

Commodore

WORLD

N.º 1, FEBRERO 1984

275 PTAS.

**REGALOS
PREMIOS**
para todos
ver Editorial pag. 4

Encuentros en tercera fase con el C-64 y el VIC-20

Una Excursión en Basic
Programas
Soft para el 700
Club Commodore
Concursos



DESCRIPCIÓN ALFABÉTICA DE LOS MNEMÓNICOS DEL 6502/6510 (I)

ADC

Suma la memoria al acumulador con acarreo

Operación: $A + M + C \rightarrow A, C$

(Ref.: 2.2.1)

N Z C I D V
V V V - - V

Modo de Desc.	Formato en ensamblador	Código Operación	Num. Bytes	Num. Ciclos
Inmediato	ADC \rightarrow Oper.	69	2	2
Pág. Cero, X	ADC Oper., X	6B	2	3
Pág. Cero, X	ADC Oper., X	75	2	4
Absoluto, X	ADC Oper., X	6D	3	4
Absoluto, X	ADC Oper., X	7D	3	4*
Absoluto, Y	ADC Oper., Y	79	3	4*
(Indir., X)	ADC (Oper., X)	61	2	6
(Indir., Y)	ADC (Oper., Y)	71	2	6*

* Suma 1 si se cambia de página.

AND

AND lógico con el acumulador

Operación: $A \wedge M \rightarrow A$

(Ref.: 2.2.3.6)

N Z C I D V
V V - - -

Modo de Desc.	Formato en ensamblador	Código Operación	Num. Bytes	Num. Ciclos
Inmediato	AND \rightarrow Oper.	29	2	2
Pág. Cero, X	AND Oper., X	2B	2	3
Pág. Cero, X	AND Oper., X	35	2	4
Absoluto, X	AND Oper., X	2D	3	4
Absoluto, Y	AND Oper., Y	3D	3	4*
(Indir., X)	AND (Oper., X)	21	2	6
(Indir., Y)	AND (Oper., Y)	31	2	6

* Suma 1 si se cambia de página.

ADC

Cambia el bit izquierdo

Operación: $C \leftarrow 7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \ 0 \rightarrow B$

(Ref.: 10.2)

N Z C I D V
V V V - - -

Modo de Desc.	Formato en ensamblador	Código Operación	Num. Bytes	Num. Ciclos
Acumulador	ASL A	0A	1	2
Pág. Cero, X	ASL Oper., X	06	2	5
Pág. Cero, X	ASL Oper., X	16	2	6
Absoluto, X	ASL Oper., X	0E	3	6
Absoluto, X	ASL Oper., X	1E	3	7

ASL

ASL

BCC

Salta si acarreo desactivado

Operación: Salta si $C = 0$

(Ref.: 4.1.1.3)

N Z C I D V
- - - - -

Modo de Desc.	Formato en ensamblador	Código Operación	Num. Bytes	Num. Ciclos
RELATIVO	BCC Oper.	90	2	2*

* Suma 1 si el salto va a la misma página.

* Suma 2 si se salta a otra página.

BCC

CONTENIDO	PAG
PERSONALIZAR TU VIC Matemáticas, ciudades, músicos, etc., se podrán beneficiar de la capacidad gráfica del VIC.....	6
SEGUIR EL VELOZ RITMO DEL VIC Forma de que los Estados del VIC sean más fáciles de leer.....	13
CLUB COMMODORE Editor de Sprites.....	17
Dos programas: Gráficos VIC + Superexpander.....	22
Carótica.....	26
Programa para hacer Scroll.....	28
Utilizando el Port.....	29
MAGIA Usos métodos fantásticos para aprovechar al máximo sus sistemas.....	32
GALERIA DE SOFT CBM-8 700. Software para el 700 El aspecto más importante en todo ordenador es el software, el 700 no es la excepción.....	35
EL RINCON DEL VICOLAGE Comunicación serie mediante el bus RS-232C. Una interesante serie de artículos que derrama billones de fotones sobre los rincones más oscuros del interface serie (RS-232).....	37
CLAVE Una Excursión al Basic. Más allá del manual Primero de una serie de artículos que le ayudará a navegar por los rápidos y los recónditos de programación Basic.....	40
FICHEROS EN DISCO Técnicas de acceso directo. Aquí, vamos a ver cómo se distribuyen los ficheros en el diskette.....	44
VIDEOCASINO Un programa original que vd. puede usar y disfrutar con su micro.....	47

PROXIMO NUMERO

- Disfrutando con las matemáticas (VIC-20 y C-64)
 - Conversión del Soft del VIC-20 al C-64
 - Ventanas CBM
 - Up Periscope para entender valores hexadecimales y binarios
 - DISK-O-VIC programa para añadir 13 nuevos mandatos a tu VIC
 - Juegos
 - Club Commodore todas vuestras aportaciones
 - Paquetes de Soft
 - Novedades
- y... MUCHAS GAFAS con comentarios y sugerencias

¡OS ESPERAMOS!

Commodore World es publicado en colaboración entre Microelectrónica y Control-Commodore y SIMSA

EQUIPO

Manuel AMADO; Adela LOPEZ; María LOPEZ; Juan MARTINEZ;
Pere MASATS; Jeffrey MILLIS; Rafael NAVARRO; Fernando M. RODRIGUEZ;
Manuel SANS; Jordi SASTRE; Valerie SHANKS...
...Y NUESTROS LECTORES

SIMSA

Coordinador María López
Pedro Megaraza, 4-8ª B
Madrid-16
Teléfono (91) 259 54 78

MICROELECTRONICA Y CONTROL-COMMODORE

Coordinador Pere Masats
Taquígrafo Serra, 7-5ª
Barcelona 29
Teléfono (93) 250 51 83/82

EDITORIAL

Nueva Etapa para los Lectores Nuevos y Bienvenida a los Viejos

La primera de todo —Bienvenidos todos al más amplio mundo de Commodore, Commodore World. Especialmente saludos a los leales del Club Commodore.

La revista nuestra y vuestra Club Commodore ha crecido tanto, y con tanto éxito, que se hace imprescindible su mayor profesionalización. Por esta razón, llegó a un acuerdo con la empresa editorial SIMSA para que se hiciera cargo de su publicación.

Esta asociación con SIMSA no va a representar, en ningún momento, un abandono de Club Commodore por parte de Microelectrónica y Control. Todo lo contrario.

Microelectrónica continuará escribiendo las secciones familiares a los "viejos" lectores y podrá añadir material nuevo de gran interés.

La sección de Club, como tal, vivirá colaboraciones, sugerencias, propuestas, participaciones, cartas, se potenciará al máximo con la mayor asistencia técnica.

SIMSA, por su parte, aportará su equipo de profesionales en el periodismo informático y con eso, está dicho todo.

Porque pretendemos que esta sea un auténtico Club y una gran familia, en nuestro equipo editorial (página 5) no se especifican competencias sino **equipo**, porque tanto los colaboradores de Microelectrónica y Control como los de SIMSA van a trabajar juntos en un trabajo de auténtico equipo.

Para lectores "viejos" y nuevos ofrecemos a continuación una breve historia de Club Commodore y el nacimiento de su segunda época como Commodore World.

BREVE RESUMEN

En septiembre de 1982, un equipo de profesionales de Microelectrónica y Control —importador exclusivo de los productos Commodore en España— creó el número cero de Club Commodore con 8 páginas. Estas 8 páginas se publicaban en el suplemento *Índice de Revista Española de Electrónica*, de forma separada mediante la adición de unas tapas se distribuyó a los suscriptores. Pronto se vio la necesidad de aumentar el volumen de la revista y, a partir de diciembre de 1982, Club Commodore se convirtió en un adosado de 16 páginas, con un número de suscriptores en continuo crecimiento y, en abril del 83, apareció por primera vez con 20 páginas. Durante toda la primera época, *Revista Española de Electrónica* ha estado incluyendo en su contenido 8 de las páginas de Club Commodore.

Microelectrónica y Control ha procedido con Club Commodore —sueto principio— descomen que consideramos de gran importancia.

1. Contribuir a la difusión y comprensión de los **ordenadores personales** en nuestro país, manteniendo al usuario informado de las novedades que van apareciendo y avanzan que se van realizando en este campo y —sobre todo— ayudándole a entender y utilizar más y mejor su equipo, haciéndole el "mismo juego".

2. Servir de medio de comunicación entre los propios usuarios, creando un auténtico **club de amigos de todos los países**, poniendo nuestras páginas a disposición de todos nuestros lectores donde puedan comunicarse entre sí, inter-

cambiando ideas, programas, opiniones, sugerencias y —que incluso— puedan llegar a conocer si lo desean.

Club Commodore, una importantísima sección de Commodore World, a partir de hoy, debe ser obra del propio usuario, miembro autónomo del Club, para ayudar —en la medida de sus fuerzas— a impulsar el avance y educación de la microinformática en España.

Ningún esfuerzo queda perdido —todo colaborador de nuestro Club se convierte automáticamente en alumno y profesor a la vez—. Desde el programa más técnico al más simple, desde la sugerencia más trivial a la más elemental, todas nuestras colaboraciones son vitales.

La ideología de SIMSA, es exactamente la misma —nacón por la que fue fundada por Microelectrónica y Control para edificar su mayoría de edad de Club Commodore.

PRESENTE Y FUTURO

Muchos son los proyectos que existen para el Club Commodore —al que repetimos, pertenecen todos los orgullosos propietarios de un equipo— en principio explicaremos que ya existe con las acciones correspondientes al integrarse en Commodore World.

En segundo lugar, os daremos una idea de algunos de los proyectos que tenemos "en mente" para el futuro, pero no un futuro a largo plazo sino a uno que se habla a la vuelta de la esquina, un futuro que es casi un ya.

PRESENTE

Colaboraciones de todo tipo —programas y artículos de interés (ver **Intercambiando experiencias**, página 16).

b) Trucos y sugerencias (ver **Magia**, página 32).

c) MarketClub (página 34).

d) Los programas que se envían —que deben venir todos con cinta o disco se devuelven lo antes posible, con algún programa de interés.

e) A partir de ahora se organizarán tres grupos de colaboradores:

1.—de 8 a 12 años.

2.—de 13 a 18 años.

3.—de 19 años en adelante.

INDICE DE ANUNCIANTES

	Página
BM.....	36
COMMODORE.....	14 y 15
EAF.....	41
ICOSA.....	26
MICROSISTEMAS.....	31
TELE SANT JUST.....	12
SAKATI.....	5
VIC 20.....	52
MARKETCLUB.....	34

Para cada grupo de edad, se sortearán cada 8 meses, 6 paquetes de software entre todos los artículos publicados.

A final de año, en Sevilla que se celebrará, se darán 10 premios especiales a las tres mejores colaboraciones de cada grupo de edad.

Entre todos los colaboradores a "Magia" se sortearán anualmente 3 "sorpresas mágicas".

¡Todas las colaboraciones deben venir escritas a máquina —a doble espacio— y se le es posible así como nuestras colaboraciones más sencillas, pueden enviarse escritas a mano con letra muy clara (Charvato para uno sirve cuando nuestro "profesor" se enfada porque los deberes son un "charvato").

Incluir información detallada del funcionamiento del programa, tanto por la que responda a ejemplos de utilización (para comprobar su funcionamiento), como el detalle de lo que hace cada línea o comando de líneas.

¡Envidia, de no tener alguna objeción personal, vuestra dirección y la telefonía para que podamos poneros en contacto unos con otros.

¡Todas las colaboraciones deben enviarse a: Pte. Maxima, Microelectrónica y Control Commodore, Vaquegráfico Soria, 7, 3.º - BARCELONA-29 antes del 10 de cada mes.

FUTURO

Desde ya, próximo número de marzo, enviad vuestras cartas con opiniones, dudas, consultas, cosas que queráis ver en la revista a nuestra redacción: Commodore World, María López Morán, Pedro Magariño, 4-8º B - MADRID-16.

¡En los diversos Clubs de Commodore que existan en el país, enviaremos detalles sobre los temas a la redacción para poder organizar entre todos alguna "junta" (En esto hablaremos en próximos números).

c) Bolsa de trabajo.—Para nuestros informáticos que estén buscando, gratis. Para las empresas que necesiten personal 300 pesetas por línea (a redacción).

UN NOMBRE

COMMODORE WORLD fue elegido entre todos nosotros porque significa EL MUNDO DE COMMODORE —un mundo internacional y grande—. Tenemos el orgullo de ser la primera revista dedicada a Commodore en Europa (aparte de Inglaterra donde hay unas cuantas) y también la primera de habla española en el mundo. (Nos están llegando pedidos de Sudamérica).

Y ACABAMOS

No sin antes expresar nuestro agradecimiento a *Revista Española de Electrónica* que tanto ayuda prestó a Club Commodore en su primera etapa y a Carlos Domercq, Presidente de Microelectrónica y Control —ya por parte de Commodore en España— por el continuo apoyo y dedicación, tanto en la primera etapa como en la segunda.

Con estas palabras sobre Club Commodore y Commodore World damos un último farolito un brío de alegría a todos nosotros.

El equipo

Personalizar tu VIC con gráficos de juegos

Animo, amantes de los juegos de los ordenadores personales! Incluso el VIC no ampliado puede proporcionar una pantalla de alta resolución y caracteres programables por el usuario para competir con las máquinas de juegos.

Los programas de juegos no son los únicos que se benefician de la capacidad gráfica del VIC. Los matemáticos, estudiantes de idiomas extranjeros y músicos la pueden utilizar para crear caracteres específicamente pertenecientes a su campo. Símbolos como Σ (sigma), λ (lambda), Ω (ohm), \sim (tilde), -- (umlaut), ^ (cuarto de tono) y ^ (clave de sol) pueden ser creados y utilizados por programadores que antes no se podían expresar dados las teclas limitadas del VIC.

El Listado 1 presenta una "hoja electrónica" para que experimentes y crees tus propios caracteres. Pero antes de hablar de los programas tenemos que explicar, aunque sea por encima, la naturaleza de los caracteres y la pantalla del VIC.

Caracteres de puntos

Cada carácter en la pantalla del VIC realmente se compone de 64 pequeñas unidades llamadas píxeles (puntos). Lo mismo pasa con todos los caracteres, desde la letra A a un símbolo gráfico invertido. Cada carácter tiene una anchura de ocho píxeles y una altura de ocho píxeles. Estos puntos pueden encenderse (claro) o apagarse (oscuro). La colocación de los píxeles oscuros en relación a los píxeles claros presenta la ilusión del carácter entero.

La memoria del VIC tiene un método especial de controlar caracteres y de saber cuáles son los píxeles que se encuentran encendidos y los que se encuentran apagados. Este proceso se llama "bit mapping" (correlación de bits), en la cual cada píxel se representa por un bit determinado en la memoria.

A lo mejor ya sabes que toda la información almacenada en el ordenador se encuentra en la forma más básica, el lenguaje binario. Los números binarios se componen sólo de los dígitos binarios (bit) 1 y 0. Este sistema es la forma más sencilla de comunicación; una representación numérica de "on" (encendido) y "off" (apagado); si y no; oscuro y claro. Los bits se agrupan en unidades de ocho que se llaman bytes, en las ubicaciones de memoria del ordenador. La cantidad de memoria que posee el ordenador se expresa en K para kilobytes (1K es igual a 1024 bytes, ó 8192 bits), representando la cantidad de información que es capaz de contener.

Los bits juegan un papel muy importante en la creación de los símbolos en el VIC. Dado que los bits se pueden encender o apagar, se utilizan directamente en la memoria del VIC para representar el estado de cada píxel. Dado que cada posición de memoria almacena un byte, una fila entera de ocho píxeles se puede almacenar en una posición de memoria en forma de un número binario. Por lo tanto, se necesitan ocho de estas posiciones para almacenar los bytes para un carácter entero.

```

0 0 1 1 1 1 0 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 0 1 0 0 1 0 0
1 1 0 0 0 0 1 1
0 0 0 0 0 0 0 0
    
```

La Figura 1a. Representación binaria del carácter ohm.



La Figura 1b. Representación de píxeles del carácter ohm.

VIC-20



Liurade 1

```

1 PRINT "[SHIFT CLR]"
2 PRINT "[CTRL I]"
10 POKE 32,28:POKE 56,28:CLR
15 FOR A=7168 TO 7680:POKE A,PEEK(A+25600):NEXT
19 POKE 30869,255
20 FOR A=7384 TO 7391: READ B:POKE A,B:NEXT
25 DATA 0,0,0,0,0,0,0,0
30 FOR A=7392 TO 7399: READ C:POKE A,C:NEXT
35 DATA 255,255,255,255,255,255,255,255
100 FOR G=7753 TO 7759:READ H:POKE G,H:NEXT
105 DATA 16,18,15,7,18,1,6
110 FOR G=38473 TO 38479:READ H:POKE G,H:NEXT
115 DATA 6,6,6,0,0,0,0,0
116 FOR G=7456 TO 7463:READ H:POKE G,H:NEXT
117 DATA 255,0,0,0,0,0,0,0
118 FOR G=7464 TO 7471:READ H:POKE G,H:NEXT
119 DATA 255,128,128,128,128,128,128,128
120 PRINT SPC(99):"[CTRL I]"UP ARROW"]="DARK"
121 FOR G=7472 TO 7479:READ H:POKE G,H:NEXT
122 DATA 128,128,128,128,128,128,128,128
125 PRINT TAB(5):CHR$(38):TAB(12):"[CTRL I]"LIGHT"
130 FOR J=1 TO 8
135 FOR G=1 TO 4
140 PRINT TAB(1) CHR$(37):
145 NEXT G
146 PRINT CHR$(38)
150 NEXT J
155 FOR G=1 TO 4
160 PRINT TAB(1) CHR$(36):
165 NEXT G
170 PRINT TAB(5) CHR$(37):
175 FOR G=1 TO 2
180 PRINT TAB(6) CHR$(38):
185 NEXT G
186 PRINT TAB(8) CHR$(36)
190 PRINT
195 FOR G=7400 TO 7407:READ V:POKE G,V:NEXT
200 DATA 28,16,16,16,16,16,16,16
205 FOR G=7408 TO 7415:READ W:POKE G,W:NEXT
210 DATA 8,20,16,16,16,16,16,82
255 PRINT"DO ONE ROW AT A TIME"
260 FOR G=1 TO 4000:NEXT
265 PRINT"START WITH TOP ROW"
270 FOR G=1 TO 4000:NEXT
271 FOR G=8076 TO 8163:POKE G,32:NEXT
275 INPUT"[CRSR UP]([CRSR UP]1ST ROW":A$
278 I=0
279 X$=A$
280 GOSUB 495
285 I=I+B+C+D+E+F+G+H
290 PRINT "[CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR
UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]
R UP]([CRSR RT)":A$
295 FOR Z=8032 TO 8119:POKE Z,32:NEXT
300 INPUT "[CRSR DN]([CRSR DN]([CRSR DN]([CRSR DN]([CRSR
DN]([CRSR DN]([CRSR DN]([CRSR DN]([CRSR DN]([CRSR DN]
R DN":B$
304 X$=B$
305 GOSUB 495
310 J=A+B+C+D+E+F+G+H
315 PRINT "[CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR
UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]
R UP]([CRSR RT)":B$
320 FOR Z=8076 TO 8119:POKE Z,32:NEXT
325 INPUT "[CRSR DN]([CRSR DN]([CRSR DN]([CRSR DN]([CRSR
DN]([CRSR DN]([CRSR DN]([CRSR DN]([CRSR DN]([CRSR DN]
R DN":C$
330 X$=C$
335 GOSUB 495
340 E=A+B+C+D+E+F+G+H
345 PRINT"[CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR
UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]([CRSR UP]
R UP]([CRSR RT)":C$

```

La Figura 1 es una representación gráfica del carácter Ω (ohm). El símbolo ohm no es un carácter estándar del VIC, pero lo utilizaremos como ejemplo de cómo se diseña un carácter personalizado.

[Create an Account](#)

Se podría diseñar el carácter usando una hoja de papel cuadrado, o se podría emplear el programa de la hoja electrónica Prograf (Listado 1). Utilizando Prograf, los caracteres pueden ser manipulados y desarrollados y se ven en seguida de qué forma se pueden representar en la pantalla. Cualquiera que sea el método utilizado, se realiza la traducción de un diseño en binario escribiendo un cero para un espacio en blanco y un uno para un espacio oscuro. Si se requiere la forma gráfica invertida, se realiza esta operación al revés, colocando un uno en un espacio en blanco y un cero en un espacio oscuro.

Con el VIC no se puede introducir la información de caracteres directamente en forma binaria, sino que se realiza mediante su equivalente decimal. Como no es una habilidad común el saber los números binarios y los equivalentes decimales, la Figura 2 se presenta aquí para que la conversión resulte más fácil.

Una forma de calcular el equivalente decimal de la información en forma de carácter binario se presenta en la "Guía de referencia del Programador del VIC". Con este método hay que sumar ocho números diferen-



La Figura 2. La posible combinación de colocación para cuatro píxeles. El valor de la derecha (para los cuatro píxeles de la derecha) más el valor de la izquierda (para los cuatro píxeles de la izquierda) es igual al valor de datos para el bote entero.


```

350 FOR Z=8076 TO 8119:POKE Z,32:NEXT
355 INPUT "[(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN)
[(CRSR DN) [(CRSR DN) [(CRSR DN) 4TH ROW":D$
358 XS=D$
359 GOSUB 495
360 L=A+B+C+D+E+F+G+H
361 PRINT "[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP
[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR RT)":D$
362 FOR Z=8076 TO 8119:POKE Z,32:NEXT
365 INPUT "[(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN
[(CRSR DN) [(CRSR DN) [(CRSR DN) 5TH ROW":E$
366 XS=E$
368 GOSUB 495
369 M=A+B+C+D+E+F+G+H
370 PRINT "[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP
[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR RT)":E$
372 FOR Z=8076 TO 8119:POKE Z,32:NEXT
374 INPUT "[(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN
[(CRSR DN) [(CRSR DN) [(CRSR DN) 6TH ROW":F$
375 XS=F$
376 GOSUB 495
377 N=A+B+C+D+E+F+G+H
378 PRINT "[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP
[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR RT)":F$
379 FOR Z=8076 TO 8119:POKE Z,32:NEXT
380 INPUT "[(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN
[(CRSR DN) [(CRSR DN) [(CRSR DN) 7TH ROW":G$
381 XS=G$
382 GOSUB 495
383 O=A+B+C+D+E+F+G+H
384 PRINT "[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP
[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR RT)":G$
385 FOR Z=8076 TO 8119:POKE Z,32:NEXT
386 INPUT "[(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN) [(CRSR DN
[(CRSR DN) [(CRSR DN) [(CRSR DN) 8TH ROW":H$
387 XS=H$
388 GOSUB 495
389 P=A+B+C+D+E+F+G+H
390 PRINT "[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP
[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR RT)":H$
391 FOR Z=8076 TO 8119:POKE Z,32:NEXT
395 PRINT "[(CRSR DN) DATA":I;J;K;L;M;N;O;P
400 POKE 7416,I:POKE 7417,J:POKE 7418,K:POKE 7419,L:POKE
7420,M:POKE 7421,N
401 POKE 7422,O:POKE 7423,P
405 PRINT "[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP)
[(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP) [(CRSR UP)":TAB(
14):CHR$(95)
415 PRINT "[(CRSR DN)":TAB(12):"CHOOSE: ",TAB(12):"1=MODI
FY",TAB(12):"2=ANOTHER",TAB(12):
418 PRINT "3=END"
420 GET Z:IF Z$="" THEN 420
422 IF Z$="1" OR Z$="I" THEN 420
425 IF Z$="I" THEN 435
427 IF Z$="2" THEN 1
430 IF Z$="3" THEN 450
435 FOR Z=8054 TO 8163:POKE Z,32:NEXT
436 CLR
437 INPUT "[(CRSR DN) [(CRSR DN) 1ST ROW":A$
438 GOTO 278
450 PRINT "SHIFT CLR:"
451 POKE 644,88
452 PRINT SPC(112)"THE END"
455 END
495 A=0:B=0:C=0:D=0:E=0:F=0:G=0:H=0
500 IF MIDS(X$,1,1)=CHR$(92) THEN A=128
501 IF MIDS(X$,2,1)=CHR$(92) THEN B=64
502 IF MIDS(X$,3,1)=CHR$(92) THEN C=32
503 IF MIDS(X$,4,1)=CHR$(92) THEN D=16
504 IF MIDS(X$,5,1)=CHR$(92) THEN E=8
505 IF MIDS(X$,6,1)=CHR$(92) THEN F=4
506 IF MIDS(X$,7,1)=CHR$(92) THEN G=2
507 IF MIDS(X$,8,1)=CHR$(92) THEN H=1
508 RETURN

```

tes —uno para cada columna de píxel. El valor de cada columna es el número 2 elevado a la potencia de dicho número de columna (enumeradas de 7 a 0, de izquierda a derecha). Este método resulta incómodo y da más trabajo que el necesario.

El método alternativo presentado en la Figura 2 demuestra todas las posibles combinaciones de colocación para cuatro píxeles. La Tabla demuestra los equivalentes decimales de los cuatro píxeles a la extrema derecha y la extrema izquierda. Utilizando este sistema, se suman los valores decimales que corresponden a los cuatro píxeles de la extrema derecha y los de la extrema izquierda de una fila de caracteres.

Tomando como ejemplo el carácter ohm, se obtiene el valor decimal del primer byte buscando el valor de los bits de la derecha (0000), 12, y sumándolo al valor de los bits de la izquierda (0000), 48. En la fórmula $D+I=P$, el valor de la derecha, D, más el valor de la izquierda, I, es igual que el valor decimal, P, para la fila entera. Se sustituye el valor correspondiente de la tabla, $12+48=60$, el valor de la fila 1.

Por lo tanto, en la fila 2, $2+64=66$. Las filas de 3 a 5 tienen el mismo valor que la fila 2, dado que los formatos de los bits son idénticos. La fila 6 tiene el valor decimal de 36 ($4+32$). El valor de la fila 7 es de 195 y la fila 8 tiene un valor de 0. Cada uno de los números decimales obtenidos mediante este proceso puede hacer un "Poke" en la memoria del VIC, y así se obtiene la información necesaria para imprimir el carácter ohm en la pantalla.

Espacio de Memoria

Antes de almacenar un carácter se tiene que preparar un sitio en la memoria que sirva de almacenamiento. El juego de caracteres que se utiliza cuando el VIC se enciende al principio se almacena en el generador

EQUIVALENCIA ESPAÑOLA

120	DARK = OSCURO
125	LIGHT = CLARO
255	PRINT "HAZLO LINEA POR LINEA"
365	PRINT "IMPRIME POR LA LINEA INICIAL"
275	1ST ROW = PRIMERA LINEA
380	2ND ROW = SEGUNDA FILA
325	3RD ROW = TERCERA FILA
155	4TH ROW = CUARTA FILA
165	5TH ROW = QUINTA FILA
374	6TH ROW = SEXTA FILA
180	7TH ROW = SEPTIMA FILA
386	8TH ROW = OCTAVA FILA
385	DATA = DATOS
405	CHOOSE = ELEGIR, MODIFY = MODIFICAR, ANOTHER = OTRO
408	END = FIN
437	1ST ROW = PRIMERA FILA
452	THE END = FIN

de caracteres ROM. El segundo juego, obtenido al pulsar las teclas "SHIFT" y Commodore a la vez, también se almacena en ROM. Dada la misma naturaleza de ROM (Memoria sólo para Lectura), no acepta ningún carácter personalizado. Sin embargo, el área RAM se encuentra totalmente disponible para su uso.

Existen dos posiciones en RAM que funcionan como punteros que le indican al ordenador dónde se consigue la información de caracteres. Estas dos posiciones se pueden modificar para que apunten a un área en la RAM del VIC no ampliado. Así, se puede colocar cualquier carácter en RAM, y el VIC podrá tener acceso.

Surge un problema cuando los punteros de caracteres se modifican para apuntar hacia RAM: se pierde la capacidad de utilizar el juego de caracteres pre-programado del VIC. Pero esto no resulta tan problemático como parece. Sólo se necesitan unas pocas líneas de programa para programar una sección de RAM con la información de caracteres contenida en ROM.

```
FOR X = 7168 TO 7679
POKE X = PEEK (X+25600)
NEXT X
```

Las direcciones de RAM que reciben la transferencia de información de caracteres de representan con X. En este ejemplo, los primeros 5K de RAM se utilizan (7168 a 7679). Se puede tener acceso a más o menos de este área utilizando valores más altos o más bajos para X y cambiando 25600 a un número que, cuando se suma al primer valor de X, es igual a 32768 (7168 + 25600 = 32768), el principio de la ROM de caracteres. La segunda línea del programa hace un "PEEK" de la información de caracteres de ROM y hace un "POKE" en la posición de RAM fijada por X.

Una vez desplazado el juego de caracteres a RAM, ocurre una cosa sorprendente: El cursor desaparece de la pantalla con el resultado de que no

se pueden editar los listados de programa. Realmente no afecta la capacidad de editar pero no se distingue muy bien dónde se encuentra el cursor. Si tú sabes que el cursor se encuentra directamente debajo de la R de "READY", que aparece en la pantalla al completarse una acción, se pueden contar el número de movimientos del cursor que se necesitan para llegar a un espacio determinado.

Hay que tener cuidado al elegir la colocación del juego de caracteres en RAM. No se puede utilizar RAM ampliada porque los punteros de chip del VIC no tienen acceso a este área. Existen dos áreas adicionales que no se deben utilizar para almacenar el juego de caracteres: 0 (aquí se almacenan datos que se requieren para que el ordenador emplee Basic) y 4096 (esto es el principio del almacenamiento del programa Basic).

La parte superior del área de RAM donde se almacenan los programas Basic (4096-7679) es la mejor posición para los caracteres. Dado que hay que guardar un poco de este espacio para el programa Basic que utiliza estos caracteres, el juego de caracteres de nuestro ejemplo sólo utilizará los primeros 5K de este área.

Los 5.5K del área de almacenamiento del programa Basic de VIC se llena a partir de varias áreas de esta región a la vez. Las variables, los "arrays", los "strings" y el mismo programa Basic se escriben en las posiciones de la memoria empezando en direcciones distintas dentro de este área. Las instrucciones del programa Basic se escriben de abajo, 4096, hacia arriba. Cualquier dato que se almacena en serie se escribe en la memoria de arriba, 7679, hacia abajo a 7657.

El punto de partida y el punto final de la información de variables y "arrays" dependen del sitio en que termina el programa Basic. Si se colocara el juego de caracteres en RAM 7168-7679, y si se utilizaran los "strings" en un programa, los datos en serie se escribirían por encima de la

información de caracteres, la cual se perdería para siempre. Resulta fácil comprender la importancia de proteger el juego de caracteres después de trasladarlo a RAM.

Igual que los punteros que tienen acceso a la información de caracteres, existen punteros que le indican al VIC donde tiene que empezar a escribir los programas y los datos en serie. Estos punteros de "strings" se pueden fijar para apuntar una posición que se encuentra por debajo del juego de caracteres, así protegiendo la información de caracteres ya que no se puede escribir encima de ésta. La Figura 3 presenta un listado de los sitios en RAM donde se pueden fijar los punteros para que incluyan un juego de caracteres incluso más grande.

Caracteres Personalizados

Para demostrar estas ideas vamos a programar nuestro carácter ohm. Los datos que calculamos antes se colocarán en las posiciones RAM 7168 a 7175. Este área contiene la información para el símbolo @ cuando se acaba de trasladar el juego de caracteres.

```
5 POKE 52, 28: POKE 56, 28: CLR
10 FOR X = 7168 TO 7679: POKE X,
PEEK
(X + 25600): NEXT X
15 POKE 36889, 255
20 FOR L = 7168 TO 7175: READ D:
POKE L, D: NEXT L
25 DATA 60, 66, 66, 66, 66, 36, 195, 0
```

Las líneas 5-15 reservan la memoria, modifican los punteros de caracteres y trasladan el juego de caracteres. La conversión del símbolo @ en el símbolo de ohm (Ω) tiene lugar en las líneas 20 y 25.

Al modificar los números de posición en la línea 20 y utilizando los datos para distintos caracteres en la línea 25, el juego entero de caracteres puede ser programado de nuevo para que contenga cualquier forma, símbolo o carácter necesario. Aunque se cambie la apariencia de una tecla, esta todavía retiene la definición original. Un signo de más (+) sigue sumando números a pesar de su nueva apariencia.

Antes de modificar un carácter hay que saber las posiciones del principio y del fin de dicho carácter en RAM. Para poder hacer esto, se obtiene el código de pantalla del carácter de la guía del usuario del VIC. Para saber la dirección inicial se multiplica este número por ocho y se suma el resultado al número que representa la posición inicial del juego de caracteres en RAM. Para obtener la dirección final se suma siete a este número.

Alta Resolución

El uso de gráficos de alta resolución produce una representación en pantalla

Posición inicial para el juego de caracteres	Se hace un "POKE" 52 y 56 con los valores...	Se hace un "POKE" 36889 con los valores...	Memoria asignada para el juego de caracteres
5120	20	253	2.5K
6144	24	254	1.5K
7168	28	255	.5K

La Figura 3. Las direcciones en RAM que permiten juegos de caracteres mayores. Se hace un "POKE" en las posiciones determinadas con los valores correctos para iniciar el juego de caracteres en la posición RAM de la columna uno. (NOTA: estos valores sirven para el VIC no ampliado).

lla más clara y más profesional. También exprese la memoria. Cuando hablamos de los caracteres personalizados dijimos que el área más pequeña que se puede manipular en la pantalla del VIC es de un pixel, que representa 1/64 de un carácter entero. Para usar los gráficos de alta resolución, hay que controlar los pixels de cada carácter, y esto se hace mediante el "bit-mapping" como hemos mencionado antes.

La pantalla del VIC tiene 22 caracteres de anchura y 23 de longitud. Esto significa que la pantalla entera del VIC tiene una resolución de 176 x 184 pixels, para un total de 32.384. Cuando cada pixel se asigna a un bit individual, se requiere un total de 4096 bytes para su almacenamiento. Obviamente, esto no es posible en el VIC no ampliado. Sin embargo, es posible realizar el "bit-mapping" en una porción de la pantalla del VIC. Un área de 8 x 8 caracteres (64 x 64 pixels) requiere sólo 512 bytes de memoria para almacenar la información, y esto cae dentro de la capacidad de memoria.

En este ejemplo, queremos combinar caracteres programables y gráficos de alta resolución. Se utilizan todos los pasos detallados antes para definir los caracteres, además de los procedimientos que se requieren para preparar la pantalla para los gráficos de alta resolución. El primer paso es borrar una porción de la memoria de pantalla para usarla para el "bit-mapping". La memoria de pantalla se ubica directamente encima del área de almacenamiento del programa Basic, desde 7680 a 8191. Para borrar los bits aleatorios de este área se emplea este comando:

```
FOR S = 7168 TO 7679: POKE S,0:
NEXT S
```

Esto hace un "POKE" de los bits de cero en las ubicaciones 7168 y 7679 de la memoria de pantalla. Un repaso rápido de los valores de las direcciones de memoria indica una diferencia de 512. Esta cifra corresponde a los 512 bytes que hemos asignado para la pantalla de alta resolución.

Un programa de alta resolución requiere varias fórmulas para ubicar y controlar los pixels individuales. Se puede comparar esto a escribir una carta a un amigo. Existen varias maneras de escribir la dirección de esa persona —cada una más específica que la anterior. Tienes que saber el país, el estado, la ciudad, el apellido, el nombre de la calle, y, por último, el nombre de tu amigo.

La localización de un pixel específico se realiza con la misma precisión. El carácter donde "reside el pixel es la unidad más grande donde éste se encuentra, como el país de tu amigo.

Esta fórmula calcula el código de representación del carácter dentro de la formación de caracteres de 8 x 8.

$$C = \text{INT}(X/8)*8 + \text{INT}(Y/8).$$

C representa el número de código de representación del carácter, y X e Y son las coordenadas horizontal y vertical del pixel que se va a trasladar.

Se calcula la fila del pixel dentro del carácter mediante la fórmula

$$R = (Y/8 - \text{INT}(Y/8))*8.$$

La variable R es el número de la fila, e Y, de nuevo es la coordenada vertical.

Para calcular el número del byte, o la dirección donde se encuentra el dígito binario del pixel, se multiplica el número del carácter (C) por ocho y se suma el resultado al número de fila (R) y 7168, el comienzo de la memoria de pantalla. (Este número de la memoria de pantalla será diferente si se utiliza otra cosa que no sea los primeros 5K de RAM para el almacenamiento de caracteres). La variable T representa el número del byte.

$$T = 7168 + 8*C + R$$

Finalmente, el número del bit se encuentra utilizando la fórmula

$$I = 7 - (X - \text{INT}(X/8)*8)$$

El número del bit se representa por I, y las coordenadas horizontal y vertical son X e Y, respectivamente.

Aplicaciones de Alta Resolución

Una aplicación ideal del control de pixels de alta resolución es la visualización de ecuaciones matemáticas. Una fórmula para una línea o una curva podría surgir en el programa y sus valores de X e Y podrían ser con-

vertidos por las cuatro fórmulas de control de pixels en los términos necesarios para que la línea o curva aparezca en la pantalla de alta resolución.

De forma semejante, otras fórmulas se podrían emplear para controlar los caracteres de juegos. Cuando esta capacidad para manipular pixels se combina en la pantalla de alta resolución con caracteres especialmente diseñados para las necesidades del programador, las posibilidades no tienen límite.

Caracteres Multi-colores

El VIC es una cosa maravillosa! Los caracteres normales (al igual que los caracteres programados por el usuario) se componen de dos colores: el color específicamente elegido mediante las teclas de control y de color, y el color de la pantalla. Sin embargo, el VIC tiene una modalidad especial que permite que cuatro colores diferentes se incluyan dentro de un solo carácter. En el VIC no ampliado, esta modalidad puede introducirse haciendo un "POKE" en el registro de control de color (646).

Se modifica el valor de este registro cada vez que se cambia el color de un carácter mediante las teclas de control y de color. Si se hace un "POKE" en el registro con un número de cero a siete se da al carácter un color de negro a amarillo, por toda la gama de colores del VIC.

Los cuatro colores que componen cada carácter en la modalidad de multicolor son los de la pantalla actual, el borde y los caracteres, además de un color auxiliar que depende del número que ha hecho el "POKE" en el registro de color.

(a) 00 11 11 00		(b) SS AA AA SS	
01 00 00 10		BB SS SS CC	
01 00 00 10		BB SS SS CC	
01 00 00 10		BB SS SS CC	
01 00 00 10		BB SS SS CC	
00 10 01 00		SS CC BB SS	
11 00 00 11		AA SS SS AA	
00 00 00 00		SS SS SS SS	

(a) Pares de bits	Asignación de color	Color
00SS	pantalla	
01BB	borde	
10CC	carácter	
11AA	auxiliar	

La Figura 4. (a) Representación binaria del carácter ahh en pares de bits. (b) Representación en pantalla del carácter ahh en la modalidad de multicolor. (c) Asignación de colores al carácter ahh en la modalidad de multicolor.

La resolución de la modalidad de multi-color es la mitad de la de la modalidad normal. Es así porque los pines de la modalidad de multi-color se controlan en pares.

El orden de los dos bits en cada par decide su color. Si el par consiste en dos bits de cero (00), los dos pines correspondientes tendrán el color de la pantalla. Si el orden de los bits es de 01, el color del borde llenará esa área de pantalla. El orden 10 imprime el color del carácter, y 11 dibuja el color auxiliar. Si colocamos nuestro carácter común en la pantalla en la modalidad de multi-color, la distribución de colores sería la que se presenta en la Fig. 4.

El uso de la modalidad de multi-color disminuye tanto los caracteres que resultan difíciles de reconocer. Sin embargo, teniendo en cuenta el color que cada par representa, sería posible crear un carácter personalizado compuesto de colores en vez de una forma específica. Los bits que se requieren para hacer una tabla de colores de cuatro colores se presentan en la figura 5.

00 00 10 10	SS SS CC CC
00 00 10 10	SS SS CC CC
00 00 10 10	SS SS CC CC
00 00 10 10	SS SS CC CC
01 01 11 11	BB BB AA AA
01 01 11 11	BB BB AA AA
01 01 11 11	BB BB AA AA
01 01 11 11	BB BB AA AA
Bits	Asignación de color

La Figura 5. Orden de bits necesarios para crear una tabla de colores de cuatro colores en la modalidad de multi-color.

AND 128) encuentra la posición de la memoria de colores. Se hace un "DOKE" en la posición 16678 para fijar el color auxiliar. Se se hace un "POKE" de 14 con un color auxiliar se fija el espacio de la esquina superior de la izquierda en la modalidad de multi-color. Si se añade 1 a H y se hace un "POKE" con el resultado con un color auxiliar el siguiente espacio a la derecha se vuelve multi-color, etc.

La colocación de áreas de pantalla que se fijaron en esta modalidad se dirige según el uso dentro del programa. Podría ser que un área determinada de la pantalla en un programa de juegos sea una zona oculta de puntos adicionales. Cuando un carácter se desplaza hacia esta área, se vuelve multi-color, y el jugador sabe que se ha ganado unos puntos adicionales.

Es posible utilizar los gráficos de multi-colores dentro de un programa simplemente añadiendo una línea que hace un "POKE" en la posición 646. Sin embargo, el resultado de esto es que todo lo que se imprime en la pantalla después será de multi-color. Es posible tener mayor control sobre esta modalidad si se establecen los espacios que serán de multi-color. La fórmula $H = 17000 + 16 * (PI - 1) * 65536$

El programa Prograf

Prograf es un programa de línea electrónica que no se dependencia y que se puede volver a usar. Se dirige entre dos teclas que requieren 113 o actúan en un pinel. Cada una de las ocho filas que componen un carácter programable se colocan por separado. Al completarse las ocho filas, se representa el carácter programable en tamaño real.

Además, se representa en forma de lista la sentencia de Datos correspondiente, que comienza toda la informa-

ción necesaria para la reproducción del carácter nuevo. Finalmente, un menú proporciona una opción de modificar el carácter existente, cambiar un carácter totalmente nuevo o terminar el programa.

Prograf se divide en varias secciones diferentes. Esto permite que el programador nuevo oja a la construcción de Prograf con facilidad, pero no interfiere su función al reducir su utilidad. Las líneas 10-19 involucran los miembros del juego de caracteres a un área reservada de la memoria. Las ve de imprimirse, el título principal hace un "POKE" en la pantalla para color, refiere a continuación líneas 100-115.

Los números de carácter 92 y 93 se eligieron para representar un bit "on" (1) y un bit "off" (0). Dos caracteres adicionales (CHR94 y CHR95) son necesarios para representar las teclas como eran antes para que se representen en un menú. La formación de caracteres de ocho filas se imprime en las líneas 130-190 punto con un menú para las teclas oscuras y claras (las líneas 120 y 125).

Para facilitar la construcción de la formación de caracteres se definen específicamente tres caracteres adicionales (las líneas 116-122). Cada línea hace una pregunta y se imprime en las líneas 275-280 la sentencia de Datos. El carácter es cambiado y un menú se imprime en la pantalla (líneas 395-415).

Dado que el menú se muestra mediante una sentencia GOSUB, la información de Datos permanece en pantalla hasta que se realiza una selección del menú. Una opción de "modificar" se refiere a las líneas 435-448 para borrar los Datos y borrar las variables de la memoria. Las líneas 450-455 terminan el programa en la exitosa modalidad de multi-color. Al final, todos los cálculos para los listados de Datos, imprimidos se realizan en GOSUB, en las líneas 485-508.

Si se analiza como es debido, Prograf resulta una herramienta de programación potente que ahorra tiempo. Fue diseñado para el uso doméstico de programadores casuales, así que no incluye ninguna provisión de seguridad. La sentencia GOSUB del menú está bien protegido en contra de alguna entrada errónea, pero el resto del programa no está protegido en absoluto. Por lo tanto, si no se pulsa la tecla correcta, se se introduce en demasiados puntos o no se introduce una fila completa, se provocará un choque de Prograf.

A veces se puede recuperar totalmente si se pulsan Run/Stop y Retornar a la vez. Sin embargo, si siguen apareciendo los problemas, hay que comprobar si el listado del programa contiene algún error.

TELE división SANT INFORMÁTICA JUST

La primera tienda especializada en el VIC-20

- PROGRAMAS EN CASSETTE, DISQUETTE, etc
- IMPRESORA, MONITORES • PROGRAMAS PROPIOS
- SERVICIO TÉCNICO

INTERFACE VIC-HAM para enviar y recibir en CW y RTTY (con cualquier equipo)
Solicite más información

Calle Mayor 7 - Tel. (93) 371 7043 - SANT JUST DESVERN (Barcelona)

Seguir el Ritmo del Veloz VIC

Presentamos unos trucos para reducir la velocidad de los listados del VIC para que sean más fáciles de leer.

Si ud. toma en serio, o no tan en serio, la programación que hace en el VIC-20 de Commodore probablemente se habrá sentido frustrado muchas veces. Aunque las limitaciones del VIC son pocas, las que tiene a menudo influyen más que las ventajas que proporciona la máquina.

Uno de los inconvenientes más importantes es el tamaño de la pantalla. Esta sólo contiene 506 caracteres a la vez bajo unas condiciones normales dado que sólo dispone de 23 filas de 22 columnas. Esto da lugar a un problema con el comando de Estado (List). Ya que un listado puede dejar muchas posiciones de la pantalla en blanco, y dado que la mayoría de las líneas en un programa requieren más de una línea de pantalla, no cabe gran cosa en la pantalla en un momento determinado.

Para que el problema sea más acusado, el comando del Estado representa un programa en lo que parece ser una velocidad de la luz. Aún pulsando la tecla de control para reducir la velocidad del listado, sigue saliendo demasiado de prisa para poderlo copiar. La única alternativa parece ser la de presentar unas pocas líneas a la vez.

Aplicando los Frenos

Ahora, aquí, y por primera vez, presentamos la solución al problema del "Estado veloz". Este truco no solamente reduce la velocidad del listado, también reduce la velocidad de cualquier otra cosa que realice el VIC, proporcionando unos resultados sorprendentes.

Un aspecto agradable de este truco es que se pueden elegir entre dos velocidades: la de barrido (fallo), y la de copiar (disponible al pulsar el botón). Este procedimiento es muy sencillo. Para reducir la velocidad del VIC, se tecla:

POKE 37158,23 POKE 37159,0

¡Esos es todo! Ahora al pasar el listado de un programa, adquiere la velocidad de barrido —ni demasiado lento ni demasiado de prisa—. Esta velocidad es la más adecuada para borrar, pero es demasiado rápida para copiar.

Para reducir la velocidad a la de copiar, se pulsa cualquier tecla (excepto la de "reset") o se aprieta la tecla "shift-lock". Esta velocidad le

da tiempo de pensar antes de que la pantalla se desplace a otro juego de caracteres. Para volver a pasar la velocidad de barrido, se vuelven todas las teclas, teniendo cuidado de que la tecla "shift-lock" quede libre.

Para volver a la velocidad normal, se hace una de tres cosas: 1) POKE 37159,66. 2) mantener pulsada la tecla de "run/stop" y pulsar la tecla "reset", o 3) teclar 10/AD.

Por qué funciona

La explicación de este truco es que las posiciones de memoria 37158 y 37159 están relacionadas directamente al sistema de interrupción del VIC, el cual controla el reloj interno del ordenador tal que se tiene acceso mediante T15, T16. Estas posiciones de memoria le indican al VIC la frecuencia con que tiene que actualizar este reloj.

Al reducir el valor de la posición de memoria 37159, el VIC actualiza el reloj con más frecuencia, y por lo tanto, podrá realizar menos tareas en el mismo espacio de tiempo. Cuando tiene un valor de 66, actualiza al ritmo normal. Cuando tiene un valor de cero, actualiza con más frecuencia, y todo lo demás parece ir más despacio.

Dado que la velocidad se tiene que reducir más todavía para ajustarse a la velocidad humana las tenemos una idea de la velocidad que pueden alcanzar los ordenadores y el lenguaje de máquina, hay que ajustarlo más aun ajustando la posición de memoria 37158. A mi parecer el valor de 23 funciona mejor que cualquier otro, pero puede que ud. se encuentre más a gusto con otro valor. Hay que tener en cuenta que un incremento o una reducción del valor almacenado en 37158 no significa necesariamente un incremento o una reducción correspondientes de la velocidad del VIC.

Otra cosa para tener en cuenta es que la modificación del valor de la posición de memoria 37159 afecta la velocidad de repetición del cursor. Yo creo que un valor de 30 proporciona la velocidad adecuada para el cursor. También ocurre que esto afecta al reloj del VIC, y por lo tanto, T16 no calcula bien la hora exacta.

Para terminar, para divertirse un poco y si tiene la paciencia para hacerlo, ejecute unos programas a estas velocidades reducidas —a veces los resultados son sorprendentes.



FIJESE EN TODO LO AHORA MISMO CON U

Aprender inglés.
Cassette C-143



Jugar ajedrez.
Cartucho 1919.



Llevar la contabilidad casera.
Disket 2001



Combatir marcianos.
Cartucho 1901



Llevar control de casa.
Cassette C-130



Tocar el piano.
Cassette C-219



Llevar una agenda profesional.
Disket D-1001



Calcular integrales.
Cassette C-137



Hacer el cubo de Rubik.
Cassette C-218



Efectuar tratamiento de textos.
Cartucho C-403



Preparar tests escolares.
Cassette C-144



Jugar al póker.
Cartucho 1908



HASTA 223 CARTUCHOS Y CASSETTES. Y SEGUIMOS AMPLIANDO.

QUE PUEDE HACER NA CINTA Y UN BOTÓN.

Enseñar matemáticas a sus hijos
Cassette C-146



Programar su dieta ideal
Cassette C-136



Saber sus términos
Cassette C-217



Aprender el lenguaje del futuro
Cassette Basic



Escoja la cinta de cassette o cartucho que le interesa de entre los 223 temas existentes. Colóquelo simplemente en el VIC-20, el ordenador familiar de COMMODORE. Apriete el botón de encendido de su televisor. ¡Y ya está! Siguiendo las instrucciones que aparecerán en la pantalla, usted, su esposa o cualquiera de sus hijos podrá practicar con toda facilidad la actividad que haya seleccionado. Sin problemas. De inmediato. Y de esta forma, al mismo tiempo que puede disfrutar el VIC 20 desde el primer minuto, va introduciéndose de forma sencilla y amena en el mundo de la informática. Leyendo el manual de instrucciones del VIC 20, de regalo, y el curso BASIC, en poco tiempo será capaz de preparar sus propios programas, de inventar sus propios juegos, de buscarle todas las aplicaciones que se le ocurran. Será capaz de dominar el emocionante lenguaje del futuro. Un lenguaje totalmente imprescindible para que sus hijos puedan optar mañana a todo buen puesto de trabajo. Sea cual sea. ¡Prepárense hoy al cambio!

Este es el infinito mundo que el VIC 20 y su amplia gama de periféricos pone a su alcance. El futuro es aquí.

Y usted ya sabe que el futuro empieza hoy.

Tener una calculadora HP
Cassette C-139



GRATIS

con su VIC 20 recibirá también:

- Manual del usuario
- Un ejemplar de la revista en castellano Club Commodore

Club
Commodore

VIC 20

VIC 20



VIC-20

EL ORDENADOR FAMILIAR CON COLOR Y SONIDO.

De venta en tiendas especializadas

COMMODORE
COMPUTER

Compartiendo Experiencias entre amigos

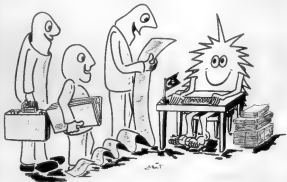
Esta sección está dedicada a la colaboración de todos nuestros lectores y está dividida en dos partes:

- 1) Programación:
Programas y similares
- 2) Magia:
Trucos, sugerencias, etc.

Habrà premios y alicientes "para todos los participantes" (ver editorial página 3).

Todas las colaboraciones deben venir escritas a máquina a doble espacio y los programas grabados en cinta. Nuestros lectores más jóvenes pueden escribir a mano pero con letra muy clara.

Enviarnos vuestra dirección para que podáis poneros en contacto unos con otros.



Editor de Sprites

COMMODORE 64



Se trata de un editor de Sprites (o BOM'S). Dispongo únicamente del manual en inglés y por lo tanto mis referencias del manual están hechas al mismo.



José Ramón Lasa Urzelai
MATXIATEGI, 34-4º A
BERGARA
Guipúzcoa
COMMODORE 64

Utilizando este editor podemos hacer el dibujo (sprite o BOM) en la pantalla (40x25) por medio de un cursor, programable, usando las teclas de función para sus desplazamientos. Acto seguido podemos acceder a cualquier opción del menú principal como son: GRABAR (un archivo de datos), LEEER (un archivo), LISTAR en líneas de basic (de 1 a 60.000) y una vez realizados el/s sprite/s podemos borrar el programa editor conservando sin embargo todas las líneas de basic existentes (DATA) obteniéndose la ejecución del mismo, quedando así listo para escribir un programa de juego que utilice sprites, o para grabar estas líneas de sentencias DATA.

En caso de necesidad de modificar el sprite se puede recurrir a la opción 2 del menú principal: CORREGIR.

NOTA: En diversas pruebas que he realizado, no sé debido a qué, una vez borrado el programa (dejando únicamente las sentencias DATA) al intentar cargar las sentencias DATA, previamente grabadas con SAVE..., he obtenido resultados negativos en algunas ocasiones. Tras la desactivación del C64 he intentado de nuevo y en este caso sí lo he conseguido.

OTRA nota: El programa está escrito en un IV de B/N y por eso que las referencias son BLANCO y NEGRO. Fondo blanco; fondo negro, línea blanca..., etc.

Después de hacer RUN aparece el menú principal con sus 7 opciones.

Pulsando 1 aparecerá en la pantalla la pregunta: Fondo blanco o negro? Una vez respondida, el programa dibuja en pantalla el tablero (fondo blanco o negro) y pasa al menú secundario con otras siete opciones.

Inicialmente pasa al modo ZONA NEGRA y queda a la espera de las acciones del cursor o modos del mismo.

En el modo ZONA (b/n) el despla-

zamiento del cursor no deja huella.

Una vez posicionado el mismo en la posición deseada se pulsa la barra espaciadora y acto seguido (en la zona de dialogo siempre) se nos pregunta por las coordenadas horizontal y vertical respectivamente. Presa respuesta, el cursor va escribiendo un rectángulo (cuadrado) cuyo vertice superior izquierdo coincide con la posición del cursor en el momento de pulsar la barra espaciadora. Dicho de otra forma el cursor se mueve hacia abajo y hacia la derecha según los desplazamientos dados y escribiendo.

En el modo LINEA (b/n) el cursor en su desplazamiento con las teclas correspondientes va imprimiendo una línea.

En el modo PUNTO (b/n) con las teclas de desplazamiento del cursor vamos posicionando el mismo en los diferentes puntos que nos interesen. En el momento que deseemos que aparezca el punto donde está situado el cursor deberemos pulsar la barra espaciadora.

Todo esto en lo referente a las acciones del cursor. Si en su lugar, o sea, si pulsamos cualquier tecla de modo que cambiamos automáticamente. Estas son las teclas numéricas de 1 a 6.

Si pulsamos el asterisco pasa el programa a ejecutar una subrutina que va transformando en bytes los contenidos de las posiciones del tablero correspondientes para acto seguido pasar al menú principal previa presentación en la pantalla del sprite dibujado usando el sprite n° 3 y expandido en los dos ejes de coordenadas.

Si en el menú principal elegimos la opción 2 inmediatamente se dibujará el tablero y acto seguido el sprite (pero

en este caso en la pantalla con definición de carácter, es decir 40x25) que en ese momento está contenido en la variable B11 a B674. Bien sea como resultado de un dibujo anterior o como una lectura de un archivo en la cima del cassette.

En la opción n° 3 podemos crear un archivo en la cima del cassette.

El programa nos preguntará el nombre del sprite a archivar y ejecutará la acción correspondiente.

Con la opción n° 4 del menú principal podemos recuperar los datos de un sprite previamente archivado en el cassette. Seguidamente podemos corregirlo o borrarlo en líneas de BASIC, es decir cualquier acción del menú principal.

En la opción n° 5 borramos el programa dejando en la memoria únicamente las líneas de basic que hayamos listado con la opción n° 6 o cualquier otra línea (de BASIC) cuyo número sea menor que 60.000. Es imprescindible que en el programa exista la línea 60.000 con un REM o con lo que sea para que trabaje perfectamente esta opción si no deberíamos cambiar en la subrutina de borrado por el número de línea a partir de la cual queremos borrar. Los punteros de comienzo de variables, matrices, etc., igualmente se actualizarán.

Con la opción número 6 podemos listar los datos del sprite en sentencias BASIC. Nos preguntará por el número de línea inicial y acto seguido pulsando en el buffer del teclado pasaran como líneas de basic con el número inicial dado por nosotros y los seis consecutivos.

En la opción n° 7 se hace END. En el caso de salir del programa por error o por parada voluntaria (tecla de stop) hay que tener en cuenta que la dimensión del buffer de teclado posición 649 queda en 1 y no en 10. Esto se debe a que al inicio del programa se limita para evitar que el cursor se nos



escape al darle repetidas veces a las teclas correspondientes ya que su desplazamiento resulta más lento que una repetición continua de las teclas.

Por lo tanto si queremos recuperar las condiciones iniciales deberemos hacer: `poke699,10 RETURN`

Los desplazamientos del cursor se hacen con las teclas de función F1-F7 de la siguiente forma:

F1	CURSOR SUBE
F3	" BAJA
F5	" IZQUIERDA
F7	" DERECHA

MENU PRINCIPAL (60.200)

Imprime en la pantalla las siete opciones de las que consta.

Hace la variable Y = 7 y pasa a la subrutina: **RESPUESTA A MENU**. De la que retorna con la tecla pulsada en la variable X. En función de ese valor se irá a la subrutina correspondiente (`ON X GOSUB...`)

MENU SECUNDARIO (60.400)

Como antes, presenta en pantalla las opciones (7).

Pasa a la subrutina **BORRADO** de la **ZONA DE DIALOGO**. Retorna si pulsamos el asterisco (*-42). Si el retorno se ha hecho de otra subrutina por haber pulsado alguna tecla numérica (1 a 6) en la sentencia `ON (R-48) GOSUB...` saltará a la subrutina del modo de cursor elegido.

AUTOMATICO (60.600)

Va a la subrutina **CURSOR** y espera allí las órdenes. Cuando retorna de la misma esta nos devuelve la variable W en las siguientes condiciones:

W=2: retoma (cambio de modo de cursor)

W=0: **MOVER CURSOR** y vuelve al comienzo (60.600)

W=1: cursor en posición. Comienza a imprimir en la pantalla:

BORRA LA ZONA DE DIALOGO
INPUT coordenadas. Las suma a las del cursor (desplazamiento) y acto seguido las firma a las dimensiones del cuadro.

Bucle **FOR-NEXT** para ir imprimiendo los puntos **ESCRIBE PUNTO RETURN**.

Esta subrutina maneja las opciones del menú secundario de zonas (**ZONA BLANCA, ZONA NEGRA**).

MANUAL (60.800)

Esta subrutina maneja las opciones: **LINEA** y **PUNTO** blanco y negro para ambas. Según los valores de la variable E que codifica el modo del cursor (E=0 será el modo **PUNTOS** y E=1 **LINEAS**) y los valores de W (diferentes teclas pulsadas) se ejecutan las acciones correspondientes.

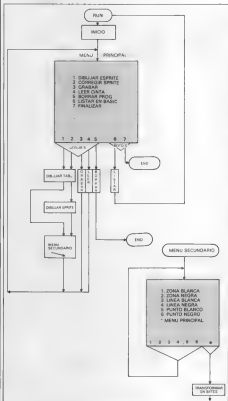
CURSOR (61.000)

Halla la posición del cursor, variable S, usando este valor el de la dirección correspondiente a la memoria pantalla.

Salta a la subrutina de **PARPADERO DE CURSOR** y después lee el teclado (**GET**) y según el valor de la tecla pulsada si la hay, da el valor correspondiente a la variable W de la siguiente forma:

Barra espaciadora: $R=32 - W=1$ y retorna.

Asterisco: $R=42 - W=2$



Si no hay ninguna, vuelve al comienzo de la subrutina.

MOVIE TITLES ARE IN ITALICS>

Según el valor de R (tecla numérica pulsada) se hace un salto ON(=132) GOSUB 1 y se pasa a la subrutina correspondiente: subir cursor, bajar, izquierda o derecha, actuando en cada caso el valor de la variable en juego (1 para desplazamientos verticales y 0 para los horizontales). Antes de reiniciar de cualquiera de ellas se limita los valores de las variables 1 y 0 a los límites de la matriz de pantalla.

SUBROUTINE 161.8(0) 62.0(0)

Codifican las variables 1 y 1, según:

los diferentes modos del cursor.

1=0 CURSOR NOT GRD

Y=1 CURSOR BLANCO

E @ EI curvor NO ESCRIBE al
desplazarse

L = 1 (1 cursor LSC RIBE) at displa-
 4356

TRANSFORMA EN BYTES (63,400)

Esta subrutina va leyendo por la sentencia `PEEK` todas las posiciones de la matriz de pantalla correspondientes al sprite (24*21), y las va agrupando haciendo el cálculo de los bytes que luego pasarán a sentencias `DATA`. Estos 63 bytes de que consta el sprite al terminar la subrutina estarán contenidos en la variable `Bil` a `B63`. Un punto negro correspondiente

a un cero y un blanco a un 1 en la
lógica binaria.

VARIABLES

L = Numero de linea de la matriz
 %4x3)

V. *Lincaea complanata* (memoria par-
tialis)

0-1 raras completas (N. de BY II.)
1. N. de BY II. dentro de la zona

11. *Acta de HOMES* (código de comercio)

100. *Calliandra* *Calliandra*

Q=8, de 10,11 actual

PROGRAMS: EDITOR DE SPORTEA

```

000000 REM *****
000010 REM *
000020 REM * EDITOR DE TEXTEIOS *
000030 REM *
000040 REM * AUTOR: *
000050 REM * JOSE RAMON LARA *
000060 REM * AGOSTO DE 1983 *
000070 REM *
000080 REM *****
000090
000100 REM: INICIALIZAR
000110 DIMO(63)
000120 P=18024:REM MEMORIA PARA ELA CARGA.
000130 P=572596:REM "L256C1" "L36C1"
COL:
000140 P=322:REM ZONA 12 MEMORIA
000150 P=532448:REM AREA DE RECIBIDO SPR.
000160 POSC=H218:
000170 REM MENI PRINCIPAL
000200
000210 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000220 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000230 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000240 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000250
000260 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000270 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000280 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000290 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000300 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000310 PRINT"EL CARGO DE LOS SPR:1 Y SPR5
HTHMBR:EL SPR1 DEL SPR1 SPR1 TDEL SPR50:3
000320
000330 VY=60000:62000
000340 REM *
000350 REM *
000360 REM *
000370 REM *
000380 REM *
000390 REM *
000400 REM *
000410 REM *
000420 REM *
000430 REM *
000440 REM *
000450 REM *
000460 REM *
000470 REM *
000480 REM *
000490 REM *
000500 REM *
000510 REM *
000520 REM *
000530 REM *
000540 REM *
000550 REM *
000560 REM *
000570 REM *
000580 REM *
000590 REM *
000600 REM *
000610 REM *
000620 REM *
000630 REM *
000640 REM *
000650 REM *
000660 REM *
000670 REM *
000680 REM *
000690 REM *
000700 REM *
000710 REM *
000720 REM *
000730 REM *
000740 REM *
000750 REM *
000760 REM *
000770 REM *
000780 REM *
000790 REM *
000800 REM *
000810 REM *
000820 REM *
000830 REM *
000840 REM *
000850 REM *
000860 REM *
000870 REM *
000880 REM *
000890 REM *
000900 REM *
000910 REM *
000920 REM *
000930 REM *
000940 REM *
000950 REM *
000960 REM *
000970 REM *
000980 REM *
000990 REM *
001000 REM *

```

```

00400 PRINT TRIM$(20) * C1 * RY$ON 261 IRY$OF 1, PU
00410 I$PC1 = TRIM$(RYS1) * RY$OF 1, ICR$(RYS1)
00420 PRINT TRIM$(20) * C1 * RY$ON 261 IRY$OF 1, RE
00430, *
004475 GOSUB 41700
00450 IF R=42 THEN GOSUB 42500:GOTO 40000
00460 ON (R=40) GOSUB 410000, 41070, 41920,
00470, 42040, 42100
00500 GOTO 60475
00510
00520
00530 REM AUTOMATIC
00540
00550
005625 RC=0
00570 GOSUB 41000
00580 IF M=1 THEN RETURN
00590 IF M=0 THEN GOSUB 41100:GOTO 60500
00600
00610 GOSUB 41700:POKE$=0.185:POKE$=1, 4
00620 INPUT "25PC INCR 320KTR, " H(H+1)
00630 INPUT "25PC INCR 320K 25PC2 " V(V+1)
00640
00650 V=L:Z=C:J=C+H+K:L=L+V+GOSUB 4170
00660 FOR C=2 TO K:L=V
00670 FOR L=L TO J
00680 GOSUB 41500
00690 NEXT L,C:Z=L+1:J=C-1
00700 REM END RETURN
00710
00720
00730 REM AUTOM
00740
00750
00760 GOSUB 41000
00770 IF M=1 THEN RETURN
00780 IF M=0 AND E=1 THEN GOSUB 41500:GO
00790 TO 41100
00800 IF M=0 AND E=1 THEN GOSUB 41100
00810 IF M=1 AND E=1 THEN GOTO 60870
00820 IF M=2 AND E=0 THEN GOSUB 41500
00830 GOTO 60070
00840
00850
00860 REM CLUSOR
00870 Z=1
00880 S=L:R=H+C+2:IF M=PEEK(S=0) M=0
00890 GOTO 60100
00900 IF C=1 IF C=H+1 THEN GOTO 61020
00910 R=R+C:IF L=INT(R/50) GOTO 60900:1000
00920
00930
00940
00950 IF R=32 THEN M=1:RETURN
00960 IF R=40 AND R=55 THEN M=1:RETURN
00970 IF R=130 OR R=133 THEN GOTO 61020
00980 RETURN
00990
01000
01010 REM END OF SUBROUTINE

```



BiQ=Valor del BYTE actual. Inicialmente se pone a cero.
K N° de BIT (0 a 7) actual dentro del BYTE.
PEEK(I)=K)- Lee el contenido de la posición de pantalla (BIT), si es blanco (160) BiQ=(BiQ)+2 (T=K), actualiza el valor

DEBUIAR EL TABLERO (62600)

Si entramos en esta subrutina desde la opción de "CORREGIR" reseta antes de pasar al menú secundario.

debido a que el valor de la variable RE en este caso es 1.

Si la entrada se hace desde la opción 1 del menú de los valores adecuados a L y a C para que el cursor aparezca en el margen superior (queriendo y asumiendo el valor adecuado a R para que al saltar al menú secundario pase al modo de cursor zona negra.

CORREGIR EL SPRITE (62600)

Esta subrutina nos hace la función inversa que la de transformar BITs, es decir, a partir de la variable Bi(i a Bi(7) va colocando todas las posiciones de la pantalla correspondientes a la matriz 24x21 con los valores

(blanco=160 o negro=32) obtenidos de la descomposición en bit's de todos los BYTES

VARIABLES

L N° de linea.
J N° de BYTE dentro de la linea.
K N° de BIT dentro del BYTE actual.
DB=Valor del byte Inicialmente DB=B(actual)
PT Posición en la matriz (dir pantalla)
PI=B/N (puntos)

GRABAR (62.600), CONTINUAR CASSETTE (63.000), LEER CINTA (63.100) LISTAR EN BASIC (63.200)

Pregunta por el nombre del sprite

```
61110 GOTO (R-1)*21: GOSUB 61200,61250,61300
61120
61130 RETURN
61140
61150
61160
61200 REM CURSOR: NEGRO
61210 L=L+1
61220 GOTO 61400
61230
61240
61250 REM CURSOR: BLANCO
61260 L=L+1
61270 GOTO 61400
61280
61290
61300 REM CURSOR: LIZO
61310 C=C+1
61320 GOTO 61400
61330
61340
61350 REM CURSOR: ORO
61360 C=C+1
61370 GOTO 61400
61380
61390
61400 REM LÍMITES DEL CURSOR
61410 IF L=23 THEN L=0
61420 IF L=0 THEN L=23
61430 IF C=23 THEN C=0
61440 IF C=0 THEN C=23
61450 RETURN
61500 REM COLOCAR PUNTO
61510 S=L*24+C+2: (32+128)*S: (POKE)S+3: 14
61520
61530 RETURN
61540
61550
61560 REM BORRADO 2, BORRADO
61570 FOR I=194 TO 240
61580 POKE I,0: NEXT I
61590
61600 PRINT "C:HOME:112:CORREGI"
61610 RETURN
61620
61630
61640
61650 REM ZONA BLANCA
61660 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)BLANCO: (R)YOSH:1"
61670 T=1: G=0
61680 GOTO 61690
61690 RETURN
61700
61710
61720
61730 REM ZONA NEGRO
61740 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)NEGRO: (R)YOSH:1"
61750 T=1: G=0
61760 GOTO 61770
61770 RETURN
61780
61790
61800 REM ZONA NEGRO
61810 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)NEGRO: (R)YOSH:1"
61820 T=1: G=0
61830 GOTO 61840
61840 RETURN
61850
61860
61870 REM ZONA BLANCO
61880 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)BLANCO: (R)YOSH:1"
61890 T=1: G=0
61900 GOTO 61910
61910 RETURN
61920
61930
61940 REM LÍNEA BLANCA
61950 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)BLANCO: (R)YOSH:1"
61960 T=1: G=0
61970 GOTO 61980
61980 RETURN
61990
```

```
61000 RETURN
61010
61020 REM LINEA NEGRO
61030 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)NEGRO: (R)YOSH:1"
61040 T=1: G=0
61050 GOTO 61060
61060 RETURN
61070
61080 REM LINEA BLANCO
61090 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)BLANCO: (R)YOSH:1"
61100 T=1: G=0
61110 GOTO 61120
61120 RETURN
61130
61140 REM LINEA NEGRO
61150 PRINT "C:CORREGI:12:SPC:1: (R)YOSH:12:SPC:1: (S)NEGRO: (R)YOSH:1"
61160 T=1: G=0
61170 GOTO 61180
61180 RETURN
61190
61200 REM BORRADO CURSOR
61210 H=POKE(S+3)
61220 IF H=32 THEN H=1
61230 IF H=160 THEN H=0
61240 GOTO 61250
61250 RETURN
61260
61270
61280 REM BORRADO CURSOR
61290 H=POKE(S+3)
61300 IF H=32 THEN H=1
61310 IF H=160 THEN H=0
61320 GOTO 61330
61330 RETURN
61340
61350
61360 REM BORRADO CURSOR
61370 H=POKE(S+3)
61380 IF H=32 THEN H=1
61390 IF H=160 THEN H=0
61400 GOTO 61410
61410 RETURN
61420
61430
61440 REM BORRADO CURSOR
61450 H=POKE(S+3)
61460 IF H=32 THEN H=1
61470 IF H=160 THEN H=0
61480 GOTO 61490
61490 RETURN
61500
61510
61520 REM BORRADO CURSOR
61530 H=POKE(S+3)
61540 IF H=32 THEN H=1
61550 IF H=160 THEN H=0
61560 GOTO 61570
61570 RETURN
61580
61590
61600 REM BORRADO CURSOR
61610 H=POKE(S+3)
61620 IF H=32 THEN H=1
61630 IF H=160 THEN H=0
61640 GOTO 61650
61650 RETURN
61660
61670
61680 REM BORRADO CURSOR
61690 H=POKE(S+3)
61700 IF H=32 THEN H=1
61710 IF H=160 THEN H=0
61720 GOTO 61730
61730 RETURN
61740
61750
61760 REM BORRADO CURSOR
61770 H=POKE(S+3)
61780 IF H=32 THEN H=1
61790 IF H=160 THEN H=0
61800 GOTO 61810
61810 RETURN
61820
61830
61840 REM BORRADO CURSOR
61850 H=POKE(S+3)
61860 IF H=32 THEN H=1
61870 IF H=160 THEN H=0
61880 GOTO 61890
61890 RETURN
61900
61910
61920 REM BORRADO CURSOR
61930 H=POKE(S+3)
61940 IF H=32 THEN H=1
61950 IF H=160 THEN H=0
61960 GOTO 61970
61970 RETURN
61980
61990
```

para luego escribirlo en un REM, y el número de línea inicial.

Escribe 7 sentencias DATA, con sus números de línea consecutivos, y con nueve datos en cada una de ellas. Finalmente escribe RUN.

Situa el cursor en la posición HOME, para que al ir golpeando en el buffer de teclado 13 vaya haciendo el mismo efecto que pulsar la tecla RETURN. Esto lo hará nueve veces: 1 REM+1 DATA+1 RUN. En la posición 198 se piden 9 N° de caracteres en buffer de teclado. Dirección inicial del buffer de teclado: 631.

BORRADO DEL PROGRAMA
163.6000

Es un número anterior de la revista

del club aparece el artículo. "Vive a las profundidades de la zona de programas". Según este formato de las líneas de BASIC usando los LINK'S vamos buscando aquella posición de la memoria a partir de la cual está el programa, o sea buscaremos la línea de programa 60 000. Una vez hallada poseamos 5 posiciones consecutivas (las cinco primeras de la línea BASIC 60 000) con cero. A continuación los punteros (174-175) Fin de programa, (43-46) comienzo de variables, (47-48) comienzo de arrays, (49-50) fin de arrays los actualizaremos a esta posición (AC). Para ello descomponemos el número decimal contenido en AC en dos:

INT (AC/256)
AC ← INT(AC/256) * 256

El motivo de haber utilizado unos números de línea tan altos (50.000) se debe a que las secuencias data no pueden numerar en una gama mayor de numeración. En el caso de que no pareciera acertada esta elección y optemos por numerar las líneas con cuatro dígitos (o sea no incluir el 6 de verde) en la línea 63,610 que sería la 3.610 deberíamos poner IF PEEK... (Número de línea primera del programa).

[illegible]

```

00110 [EQUATE CONTROL] =
00111 00111 PRINT "E2C2R2A2D12 3C3P4R3H3 3C3R4M4D3 12P2C3"
00112 PUL3P312 12P3 10R4M3 12P1 12E4L3 12P1 12C3R12R4D3 12P123
00113 12P123
00114 GET C=1:IF C#="" THEN GOTO20
00115 00120 PRINT "E1C1P1 12C1R1S1D1 12C1R1M1P1"
00116 00140 RETURN
00117 00100 0
00118 00105 0
00119 00100 REM LEER CINTA
00120 00110 INPUT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00121 00130 "E1P1" DO: [EQUATE SPRITE] = 0
00122 00120 GOSUB 42000
00123 00130 OPEN "1-1-B".MB
00124 00140 FOR I=1 TO 40
00125 00150 INPUT#1, S=C
00126 00160 NEXT I
00127 00165 CLOSE 1
00128 00170 GOSUB 42000+200
00129 00170 RETURN
00130 0
00131 0
00132 0
00133 00120 REM 12570R LINES: 4001C
00134 00121 INPUT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00135 00130 [EQUATE SPRITE] = 0
00136 00140 INPUT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00137 00150 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00138 00165 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00139 00170 FOR I=1 TO 7
00140 00174 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00141 00175 FOR I=1 TO 7
00142 00176 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00143 00177 IF C#="" THEN PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00144 00178 NEXT I
00145 00179 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00146 00180 NEXT I
00147 00181 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00148 00182 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00149 00183 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00150 00184 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00151 00185 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00152 00186 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00153 00187 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00154 00188 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00155 00189 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00156 00190 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00157 00191 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00158 00192 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00159 00193 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00160 00194 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00161 00195 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00162 00196 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00163 00197 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00164 00198 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00165 00199 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00166 00200 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00167 00201 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00168 00202 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00169 00203 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00170 00204 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00171 00205 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00172 00206 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00173 00207 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00174 00208 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00175 00209 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00176 00210 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00177 00211 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00178 00212 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00179 00213 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00180 00214 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00181 00215 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00182 00216 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00183 00217 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00184 00218 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00185 00219 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00186 00220 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00187 00221 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00188 00222 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00189 00223 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00190 00224 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00191 00225 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00192 00226 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00193 00227 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00194 00228 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00195 00229 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00196 00230 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00197 00231 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00198 00232 PRINT "E1C1R1 12C1R1D1 12C1R1M1 12C1R1H1 24R4M4D3"
00199 00233 PRINT "E1C1R1 12C1R1D1 12C
```



2 PROGRAMAS: GRAFICAS VIC 20 + SUPEREXPANDER

El primero de ellos no necesita ningún comentario, es tan sólo una vistosa "introducción" para otros programas que puede ser variado a gusto de consumidor y que demuestra las posibilidades del *Superexpander* en lo que se refiere al dibujo.



El proceso de fabricación no es muy complicado, primero se dibuja en un papel cuadrículado y se divide el papel en 1023x1023 puntos, señalando cada uno en el dibujo, así indicar que hay que dividir la componente "Y" del comando DRAW por 817 para que el dibujo quede proporcionado. Consume aproximadamente 4 kb de memoria.

El segundo programa es más interesante, porque con él podemos estudiar las funciones haciendo su representación gráfica, nos permite ampliar una parte de la gráfica por si nos interesa estudiarla con detalle y nos indica en qué partes de la curva existen asíntotas o lugares en los que la función se hace infinito.

En realidad, representar gráficamente una función no resulta difícil con el *Superexpander*. Debemos definir la función, establecer los valores de la componente "X" y los correspondientes de la "Y". Pero esto a veces, para algunas funciones no resulta tan fácil porque cuando X = 0 se hace infinito y dan error, interrumpiendo el programa, en otros casos esto ocurre cuando X es negativo.

El presente programa trata de evitar todo esto. Al ponerlo en funcionamiento, nos indica que pulsemos la tecla "F" y definamos la función a continuación, después pulsemos "RETURN" dos veces y entonces el VIC nos pregunta si deseamos establecer los límites de "X" después de responderle, nos indica que estudia la función, nos da una tabla de valores significativa y nos indica los límites establecidos para los dos ejes así como el valor de cada división del eje vertical. Después es dibujada la gráfica.

Si deseamos ver con detalle una parte de la gráfica solo tenemos que establecer los valores de la "X" de esa parte concreta.

En general el programa tiene tres partes: A) Definir y estudiar la función (hasta la línea 200). B) Representar la tabla de valores y los límites de los ejes (hasta la línea 300). C) Pinar la gráfica (de la 370 a la 405).

A —Para definir la función cómodamente definimos la tecla "F" como el comienzo de la línea 80, después se define la función y es entrada en el programa como una cadena "AS" que es igual que una línea de programa. De esta obtenemos una parte "BS" según la línea 50. La línea 65 escribe en la pantalla (en blanco, no se ven las cadenas AS y BS como si fuese una línea de programa y debajo escribe "Run 80". La línea 70 termina el programa, pero detiene el cursor encima del "4" de AS, al pulsar Return una vez se introduce la línea 80 en el programa y el cursor se detiene encima de la "R" de Run 80, y al pulsar Return otra vez continúa el programa.

Las líneas 135-150 estudian la cadena BS (la función) estableciendo límites de "X" para ciertos tipos de funciones.

Para evitar encontrarnos con error por división por cero y los valores

Guillermo Falgueras Cano
Apartado 211
LA LINEA
(Cádiz)

demandados grandes, se establece la línea 185 en la que a los valores de "X" se les va sumando incrementos (H, IS), si el valor de la función para esos valores es demasiado grande el siguiente valor es eliminado.

Las líneas 165 y 175 sirven para establecer el valor de K que será el máximo del eje "Y", también para el valor de R que será normalmente uno, salvo cuando el intervalo de "X" que hemos tomado es muy pequeño (R determina los valores en los que aumenta el eje "X").

La línea 195 determina la posición del eje vertical "KV" y la 190 la del horizonte "KF".

B) Las líneas 205 a la 295 estudian y pinta la tabla de valores y los límites de la función. Este estudio de la tabla se hace aparte de la representación.

Las líneas 300 a 365 pinta los tres las divisiones de ejes y el cuadrulado de la pantalla (este último opcional por ser útil pero engorrazoso).

C) La parte más importante del programa está en las líneas 370 a 405 que son las que pinta la gráfica. La línea 370 prepara el valor de DN, componente horizontal, que depende el primer valor X1 del número de valores a representar, P y de la dimensión de la pantalla.

La línea 380 prepara el valor de la componente vertical, DV, que depende de la función, de la posición del eje horizontal, de DN, del nº de valores, adecuándolos a los 1023 puntos horizontales y verticales de la pantalla.

Según mis observaciones, y después de estudiar cientos de funciones, el programa es capaz de representarnos casi todas, sólo falla en algunos casos



```

265 PRINT"ICLRIC7CRSRIC1RVSONJLIMITEI
1RVSOFI"
270 PRINT"ICSRSDI"X"VARIAE1SPCIENTRE" ;P
PRINTX)"/"ICSPCIVE1SPC3"/X2
275 PRINT"ICSRSDI"Y"VARIAE1SPCIENTRE" ;P
PRINTINT(AM1888+.5)/1888"/"ICSPCIVE1SPC3"/
INT(OM1888+.5)/1888
280 PRINT"ICSRSDICADRE1SPC3PUNTOE1SPC3E
H1SPC3EL1SPC3EJEC2SPC3VERTICALE1SPC3CO
RRESPONDE1SPC3IC" ;PRINT INT(K/P*1888)/188
8)"UNID,"
285 IFP3388P<18THENPRINT"ICSRSDIC3CRSR
R1E1RVSONPULSE1SPC3UNRE1SPC3TECLAE1RVSON
OF3" ;GOTO3295
290 PRINT"ICSRSDIC1RVSONJDESERE1SPC3OUR
DRICULAG(54H)1RVSOFI"
295 GETP4:IFP4=""THEN295
300 GRAPHIC2:IFP4="H"ORP3388P<18THEN338
305 FORI=1TOP
310 DRAW1,0,01TO1823,01
315 DRAW1,01,0TOD1,1823
320 01=01+1823/P
325 NEXT
330 DRAW1,0,KHTO1823,KH
335 DRAW1,0,KH-4TO1823,KH-4
340 DRAW1,KV,0TOKV,1823

```

```

345 FORI=1TOP
350 POINT0,KV,02
355 POINT0,02,KH
360 D2=D2+1823/P
365 NEXT
370 D2=1:IFX18THENHX=X1*1823/P
375 P=P/1823:K=K/P
380 D2=K-P*P*XI+P*DX*KK
385 Z=5:IFAB5<KH-FNA(X1+P+DX*2)+KK+DX
1048THENZ=1
390 IFDX>1823THENH418
395 IFDY>1823ORDY<0THENH2=DX+2*2 ;GOTO388

```

```

400 POINT1,DX,DY
405 DX=DX+2 ;GOTO388
410 GETP4:IFP4=""THENH418
415 PRINTH2:CLR
420 PRINT"ICLRIC12CRSDIC2CRSRIPARE1SP
C3JTRAE1SPC3FUNCTIONE3SPC3ICSRSDIC1ICRSP
RIPULSE1RVSONHIC1RVSOFI"
425 PRINT"ICSRSDIC2CRSRIPARE1SPC3LRI
1SPC3JTRAE1SPC3ICRVSONHIC1RVSOFI"
430 GETP4:IFP4<>"S"ANDP4<>"H"THENH430
435 IFP4="S"THENZ5
440 GOTO88

```

PROGRAM: FALGUEBAS.NHAP

```

5 GRAPHIC2
10 REGI0H0
15 DRAW1,529,0TQ493,64TQ422,64TQ404,51TQ
349,56TQ225,53
20 DRAW1,225,53TQ287,25TQ171,49TQ171,67T
Q149,64TQ113,182TQ119,128TQ119,153TQ136,
132
25 DRAW1,136,192TQ133,384TQ100,435TQ83,5
28TQ108,576TQ128,576TQ99,715TQ268,718TQ2
52,757
30 DRAW1,252,757TQ235,768TQ293,822TQ367,
795TQ311,895TQ333,768TQ369,768TQ373,795T
Q466,778
35 DRAW1,466,778TQ582,715TQ565,665TQ612,
552TQ593,435TQ632,363TQ643,356TQ643,328
40 DRAW1,643,328TQ717,386TQ772,217TQ761,
128TQ816,77TQ598,0
45 DRAW1,493,64TQ761,128
50 DRAW1,128,215TQ171,199TQ171,228TQ261,
225TQ288,448TQ216,68TQ195,704
55 DRAW1,216,68TQ278,814TQ344,537TQ454,
576TQ511,768
60 DRAW1,819,422TQ798,475TQ822,468TQ836,
486TQ858,448TQ827,435TQ819,422
65 DRAW1,823,489TQ888,388TQ899,422TQ858,
483
70 CIRCLE1,766,576,12,12
75 DRAW1,298,614TQ297,698TQ484,698TQ495,
614TQ298,614

```

```

80 DRAW1,297,648TQ485,648DRAW1,297,665T
Q485,665
85 REGI0H5:PRINT1,369,625:PRINT1,369,675
90 REGI0H0:DRAW1,648,1823TQ648,896TQ693,
832TQ1823,832
95 DRAW1,738,928TQ885,988TQ791,894TQ778,
972TQ737,928
100 CIRCLE1,844,982,21,21
105 CIRCLE1,742,975,19,19
110 CIRCLE1,658,1888,8,16
115 DRAW1,718,898TQ688,924TQ685,873TQ693,
898
120 DRAW1,928,915TQ885,968TQ908,975TQ922,
915
125 CIRCLE1,968,875,15,5
130 CIRCLE1,311,988,16,16
135 REGI0H4:PRINT1,682,58
140 PRINT1,198,512
145 REGI0H2
150 CHR15,4,"LRI1SPC3LINER"
155 FORI=1TO1588:NEXT
160 CHR3,3,"PULSE1SPC3SHIFT" ;CHRR4,0,"
V" ;CHRR5,5,"PUL/STOP"
165 CHRR18,0,"VCI1SPC3IESPERE1" ;DRAW1,0,9
58TQ588,958TQ588,1823

```


Elektrocomputer

ELEKTROCOMPUTER PRESENTA SUS NUEVOS PRODUCTOS PARA EL VIC-20 Y EL COMMODORE-64. **DATAMASTER 64** Y **CONTROLADOR - CB**, QUE AMPLIAN LAS POSIBILIDADES DE SU ORDENADOR.
DE VENTA EN DISTRIBUIDORES AUTORIZADOS DE TODA ESPAÑA.



***DATAMASTER 64** _ SOFISTICADA BASE DE DATOS PARA EL C-64 .

PENSADA PARA TRABAJAR CON LA UNIDAD DE DISCO 1541, SIENDO MUY VERSATIL APROVECHA AL MAXIMO LA CAPACIDAD DE MANIOBRA Y ALMACENAMIENTO, NUMERO DE REGISTROS VARIABLE *EJ. 5000 DE 30 CARACTERES*, FORMATEADOS Y COPIAS PROGRAMADAS, SALIDA A IMPRESORA (PARALELO CENTRONICS Y SERIE RS232) CHEQUEO OPERACIONES

DISCO . GARANTIA 3 MESES . MANUAL COMPL. EN CASTELLANO — P.V.P. 11.800' PTAS.

***CONTROLADOR - CB** _ CONTROLADOR DE 8 RELES

PARA EL VIC-20 Y EL C-64 . DE FORMA MUY SENCILLA PODEMOS HACER HASTA 255 COMBINACIONES ENTRE LOS 8 RELES, CON UN CONSUMO DE 1000W. A 220 VOLT. CADA UNO, CON LO CUAL PODEMOS ACCIONAR TODO TIPO DE LUCES O MECANISMOS . INSTRUCC. INCLUIDAS . 3 MESES GARANTIA — P.V.P. 9.800' PTAS.



VIA AUGUSTA - 120 - TEF. (93) 2180699 - BARCELONA - 6



Carotena

Es un juego provisto de niveles de velocidad y dificultad, no necesita ampliación de memoria pero requiere la utilización del Joystick.



Programa remitido por
Luis Carballo T.
Por favor, manda tu dirección

"Carotena" es una rápida, divertida y hambrienta serpiente. Ella corre por toda la pantalla con cuidado de no chocar contra los muros, los otros, devorando su comida favorita (los asteriscos), provistos de una programación especial que desarrolla su crecimiento y cuenta más comida más grande se basa.

La misión es controlar a "Caro-

tena" con la utilización de un joystick.

Después al cargar el programa, no déjen todos probados en "Carotena" ya que podrá apreciar la utilización del joystick.

JUEGOS Y PROGRAMAS SELECCIONADOS PARA COMMODORE-64 Y VIC-20

Pídanos catálogo

Gratuito

Escribiéndonos:

ICOSA
EDIFICIO TORRE
ORENSE

DESCUENTOS A DISTRIBUIDORES
DE COMMODORE



ICOSA

EXPLICACION DEL PROGRAMA

10-80 Instrucciones variables. Dimensiones. Colores al fondo y de la pantalla "gráficos" para presentación y instrucciones de juego.

90-100 Dibujos gráficos de la pantalla (tramos electrónicos).

100-160 Funciones Random para el asterisco y "Carotena".

170-200 Instrucciones para la utilización de joystick.

210 Chequeo de "Carotena" contra el muro.

240 Chequeo de "Carotena" con un asterisco.

250 Lugar del muro o asterisco.

260-320 Localización de la cola de "Carotena".

330 Sonido de "Carotena".

350 Movimiento de "Carotena".

360 Control de velocidad de "Carotena".

370-420 Presentación del juego.

430-560 Instrucciones y niveles de juego.

570 Opcion laberinto difícil.

590 Opcion laberinto fácil.

600-690 Fin de juego. Señalización de puntuación. Comienzo de un nuevo juego.

700-720 Iniciación a localización para el nuevo asterisco.

730-750 Rutina Language Machine.

760-780 Laberinto mediante asterisco como lista.



PROGRAM: CRUSTEER

```

10 DT=60:GOTOH=DT:J=0:INQ=100
20 FOR J=0 TO 5:INQ=INQ+1:POKE20+J,INQ:INQ=J
30 IF J=100:GOTOH=1:J=J+1
40 PRINT"THE LIGHT IS GREEN:POKE20+J,11:POKE
50 20+J,15:J=J+1:GOTOH=J+1:POKE20+J=J+1
60 J=0:POKE20+J=0
70 J=0:POKE20+J=0:POKE20+J=0:POKE20+J=0
80 GOTOH=J
90 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
100 J+1
110 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
120 J+1
130 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
140 J+1
150 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
160 J+1
170 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
180 J+1
190 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
200 J+1
210 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
220 J+1
230 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
240 J+1
250 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
260 J+1
270 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
280 J+1
290 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
300 J+1
310 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
320 J+1
330 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
340 J+1
350 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
360 J+1
370 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
380 J+1
390 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
400 J+1
410 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
420 J+1
430 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
440 J+1
450 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
460 J+1
470 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
480 J+1
490 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
500 J+1
510 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
520 J+1
530 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
540 J+1
550 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
560 J+1
570 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
580 J+1
590 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
600 J+1
610 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
620 J+1
630 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
640 J+1
650 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
660 J+1
670 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
680 J+1
690 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
700 J+1
710 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
720 J+1
730 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
740 J+1
750 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
760 J+1
770 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
780 J+1
790 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
800 J+1
810 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
820 J+1
830 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
840 J+1
850 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
860 J+1
870 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
880 J+1
890 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
900 J+1
910 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
920 J+1
930 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
940 J+1
950 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
960 J+1
970 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
980 J+1
990 FOR J=0 TO 5:POKE20+J=POKE20+J,1:POKE20+J=
1000 J+1

```

```

450 PRINT"THE LIGHT IS GREEN:POKE20+J,11:POKE
460 20+J,15:J=J+1:GOTOH=J+1:POKE20+J=J+1
470 GOTOH=J
480 GOTOH=J
490 GOTOH=J
500 GOTOH=J
510 GOTOH=J
520 GOTOH=J
530 GOTOH=J
540 GOTOH=J
550 GOTOH=J
560 GOTOH=J
570 GOTOH=J
580 GOTOH=J
590 GOTOH=J
600 GOTOH=J
610 GOTOH=J
620 GOTOH=J
630 GOTOH=J
640 GOTOH=J
650 GOTOH=J
660 GOTOH=J
670 GOTOH=J
680 GOTOH=J
690 GOTOH=J
700 GOTOH=J
710 GOTOH=J
720 GOTOH=J
730 GOTOH=J
740 GOTOH=J
750 GOTOH=J
760 GOTOH=J
770 GOTOH=J
780 GOTOH=J
790 GOTOH=J
800 GOTOH=J
810 GOTOH=J
820 GOTOH=J
830 GOTOH=J
840 GOTOH=J
850 GOTOH=J
860 GOTOH=J
870 GOTOH=J
880 GOTOH=J
890 GOTOH=J
900 GOTOH=J
910 GOTOH=J
920 GOTOH=J
930 GOTOH=J
940 GOTOH=J
950 GOTOH=J
960 GOTOH=J
970 GOTOH=J
980 GOTOH=J
990 GOTOH=J
1000 GOTOH=J

```

Programa para Hacer Scroll



Pere Giral Carrota
Urbanización La Tribuna, c/Guillermo, s/n, Fornells de la Selva,
Telés: 20 30 00 - 47 61 87
Girona.

COMMODORE 64

Se trata de una rutina en c/m que ocupa un total de 47 bytes y que puede ser colocada en cualquier parte de la memoria.

Su función es hacer scroll en pantalla arriba o abajo y en una zona previamente elegida por el usuario, siendo muy útil para presentación de listados, manteniendo siempre un mensaje en pantalla, por ejemplo la cabecera del listado.

Los parámetros de dicha rutina están ubicados en página cero en dos bytes libres para el usuario y son los siguientes:

M B - 251 Primera línea de scroll

M C - 252 Última línea de scroll

Siendo M el inicio de carga de la rutina en memoria SYS(M) efectúa un scroll arriba y SYS (M+25) lo hará abajo.

En el programa adjunto se ha dejado para M el valor 49152 (40000) que es un lugar que está a salvo de cualquier invasión de virus.

En dicho programa y para completarlo, después de la carga en memoria de la rutina he puesto un ejemplo en el que se hacen "scrollar" los mensajes de error de C-64 arriba o abajo según se pulse la tecla de cursor correspondiente.

Esperando que esta colaboración os pueda ser de utilidad me despido de vosotros deseándoos grandes éxitos en la labor que estáis ejecutando.

P.D. Una aclaración después de efectuar la rutina de scroll sea arriba o sea abajo el cursor queda siempre a punto para realizar el PRINT o el INPUT en la línea que ha quedado libre el scroll.

La última línea (parámetro SC252) no puede tener valor superior a 23, para evitar que el ordenador haga otro scroll por su cuenta texto solo sucede cuando el scroll es hacia arriba.

PROGRAMA: SKROLL G64RLT

```
1. REM          *SKROLL*
2. REM          "CARGA" <P.G64RLT>
3. REM
4. M=49152:REM UBICACIÓN RUTINA *C6400*
5. REM
6. REM          *CARGA RUTINA*
7. REM
8. FOR I=NTOM+46:PERDIB:POKE I,BY:NEXT
9. REM
10. REM          -----
11. DATA 166,251,32,240,273,189,241,236,13
12. DATA 101,210,32,288,233,232,228
13. DATA 152,288,238,134,214,76,255,233,16
14. DATA 32,240,273,189,239,236,133
15. DATA 172,181,216,32,288,273,282,228,25
16. DATA 200,238,240,229
17. REM          *****
18. REM
19. REM          *PROGRAMA EJEMPLO*
20. REM          -----
21. PRINT "([CLP])CARGA RUTINA *C6400*"
22. FOR I=NTOM+46:PERDIB:POKE I,BY:NEXT
23. REM
24. REM          *CARGA RUTINA*
25. REM
26. REM          *CARGA RUTINA*
27. REM
28. REM          *CARGA RUTINA*
29. REM
30. REM          *CARGA RUTINA*
31. REM
32. REM          *CARGA RUTINA*
33. REM
34. REM          *CARGA RUTINA*
35. REM
36. REM          *CARGA RUTINA*
37. REM
38. REM          *CARGA RUTINA*
39. REM
40. REM          *CARGA RUTINA*
41. REM
42. REM          *CARGA RUTINA*
43. REM
44. REM          *CARGA RUTINA*
45. REM
46. REM          *CARGA RUTINA*
47. REM
48. REM          *CARGA RUTINA*
49. REM
50. REM          *CARGA RUTINA*
51. REM
52. REM          *CARGA RUTINA*
53. REM
54. REM          *CARGA RUTINA*
55. REM
56. REM          *CARGA RUTINA*
57. REM
58. REM          *CARGA RUTINA*
59. REM
60. REM          *CARGA RUTINA*
61. REM
62. REM          *CARGA RUTINA*
63. REM
64. REM          *CARGA RUTINA*
65. REM
66. REM          *CARGA RUTINA*
67. REM
68. REM          *CARGA RUTINA*
69. REM
70. REM          *CARGA RUTINA*
71. REM
72. REM          *CARGA RUTINA*
73. REM
74. REM          *CARGA RUTINA*
75. REM
76. REM          *CARGA RUTINA*
77. REM
78. REM          *CARGA RUTINA*
79. REM
80. REM          *CARGA RUTINA*
81. REM
82. REM          *CARGA RUTINA*
83. REM
84. REM          *CARGA RUTINA*
85. REM
86. REM          *CARGA RUTINA*
87. REM
88. REM          *CARGA RUTINA*
89. REM
90. REM          *CARGA RUTINA*
91. REM
92. REM          *CARGA RUTINA*
93. REM
94. REM          *CARGA RUTINA*
95. REM
96. REM          *CARGA RUTINA*
97. REM
98. REM          *CARGA RUTINA*
99. REM
100. REM          *CARGA RUTINA*
101. REM
102. REM          *CARGA RUTINA*
103. REM
104. REM          *CARGA RUTINA*
105. REM
106. REM          *CARGA RUTINA*
107. REM
108. REM          *CARGA RUTINA*
109. REM
110. REM          *CARGA RUTINA*
111. REM
112. REM          *CARGA RUTINA*
113. REM
114. REM          *CARGA RUTINA*
115. REM
116. REM          *CARGA RUTINA*
117. REM
118. REM          *CARGA RUTINA*
119. REM
120. REM          *CARGA RUTINA*
121. REM
122. REM          *CARGA RUTINA*
123. REM
124. REM          *CARGA RUTINA*
125. REM
126. REM          *CARGA RUTINA*
127. REM
128. REM          *CARGA RUTINA*
129. REM
130. REM          *CARGA RUTINA*
131. REM
132. REM          *CARGA RUTINA*
133. REM
134. REM          *CARGA RUTINA*
135. REM
136. REM          *CARGA RUTINA*
137. REM
138. REM          *CARGA RUTINA*
139. REM
140. REM          *CARGA RUTINA*
141. REM
142. REM          *CARGA RUTINA*
143. REM
144. REM          *CARGA RUTINA*
145. REM
146. REM          *CARGA RUTINA*
147. REM
148. REM          *CARGA RUTINA*
149. REM
150. REM          *CARGA RUTINA*
151. REM
152. REM          *CARGA RUTINA*
153. REM
154. REM          *CARGA RUTINA*
155. REM
156. REM          *CARGA RUTINA*
157. REM
158. REM          *CARGA RUTINA*
159. REM
160. REM          *CARGA RUTINA*
161. REM
162. REM          *CARGA RUTINA*
163. REM
164. REM          *CARGA RUTINA*
165. REM
166. REM          *CARGA RUTINA*
167. REM
168. REM          *CARGA RUTINA*
169. REM
170. REM          *CARGA RUTINA*
171. REM
172. REM          *CARGA RUTINA*
173. REM
174. REM          *CARGA RUTINA*
175. REM
176. REM          *CARGA RUTINA*
177. REM
178. REM          *CARGA RUTINA*
179. REM
180. REM          *CARGA RUTINA*
181. REM
182. REM          *CARGA RUTINA*
183. REM
184. REM          *CARGA RUTINA*
185. REM
186. REM          *CARGA RUTINA*
187. REM
188. REM          *CARGA RUTINA*
189. REM
190. REM          *CARGA RUTINA*
191. REM
192. REM          *CARGA RUTINA*
193. REM
194. REM          *CARGA RUTINA*
195. REM
196. REM          *CARGA RUTINA*
197. REM
198. REM          *CARGA RUTINA*
199. REM
200. REM          *CARGA RUTINA*
201. REM
202. REM          *CARGA RUTINA*
203. REM
204. REM          *CARGA RUTINA*
205. REM
206. REM          *CARGA RUTINA*
207. REM
208. REM          *CARGA RUTINA*
209. REM
210. REM          *CARGA RUTINA*
211. REM
212. REM          *CARGA RUTINA*
213. REM
214. REM          *CARGA RUTINA*
215. REM
216. REM          *CARGA RUTINA*
217. REM
218. REM          *CARGA RUTINA*
219. REM
220. REM          *CARGA RUTINA*
221. REM
222. REM          *CARGA RUTINA*
223. REM
224. REM          *CARGA RUTINA*
225. REM
226. REM          *CARGA RUTINA*
227. REM
228. REM          *CARGA RUTINA*
229. REM
230. REM          *CARGA RUTINA*
231. REM
232. REM          *CARGA RUTINA*
233. REM
234. REM          *CARGA RUTINA*
235. REM
236. REM          *CARGA RUTINA*
237. REM
238. REM          *CARGA RUTINA*
239. REM
240. REM          *CARGA RUTINA*
241. REM
242. REM          *CARGA RUTINA*
243. REM
244. REM          *CARGA RUTINA*
245. REM
246. REM          *CARGA RUTINA*
247. REM
248. REM          *CARGA RUTINA*
249. REM
250. REM          *CARGA RUTINA*
251. REM
252. REM          *CARGA RUTINA*
253. REM
254. REM          *CARGA RUTINA*
255. REM
256. REM          *CARGA RUTINA*
257. REM
258. REM          *CARGA RUTINA*
259. REM
260. REM          *CARGA RUTINA*
261. REM
262. REM          *CARGA RUTINA*
263. REM
264. REM          *CARGA RUTINA*
265. REM
266. REM          *CARGA RUTINA*
267. REM
268. REM          *CARGA RUTINA*
269. REM
270. REM          *CARGA RUTINA*
271. REM
272. REM          *CARGA RUTINA*
273. REM
274. REM          *CARGA RUTINA*
275. REM
276. REM          *CARGA RUTINA*
277. REM
278. REM          *CARGA RUTINA*
279. REM
280. REM          *CARGA RUTINA*
281. REM
282. REM          *CARGA RUTINA*
283. REM
284. REM          *CARGA RUTINA*
285. REM
286. REM          *CARGA RUTINA*
287. REM
288. REM          *CARGA RUTINA*
289. REM
290. REM          *CARGA RUTINA*
291. REM
292. REM          *CARGA RUTINA*
293. REM
294. REM          *CARGA RUTINA*
295. REM
296. REM          *CARGA RUTINA*
297. REM
298. REM          *CARGA RUTINA*
299. REM
300. REM          *CARGA RUTINA*
301. REM
302. REM          *CARGA RUTINA*
303. REM
304. REM          *CARGA RUTINA*
305. REM
306. REM          *CARGA RUTINA*
307. REM
308. REM          *CARGA RUTINA*
309. REM
310. REM          *CARGA RUTINA*
311. REM
312. REM          *CARGA RUTINA*
313. REM
314. REM          *CARGA RUTINA*
315. REM
316. REM          *CARGA RUTINA*
317. REM
318. REM          *CARGA RUTINA*
319. REM
320. REM          *CARGA RUTINA*
321. REM
322. REM          *CARGA RUTINA*
323. REM
324. REM          *CARGA RUTINA*
325. REM
326. REM          *CARGA RUTINA*
327. REM
328. REM          *CARGA RUTINA*
329. REM
330. REM          *CARGA RUTINA*
331. REM
332. REM          *CARGA RUTINA*
333. REM
334. REM          *CARGA RUTINA*
335. REM
336. REM          *CARGA RUTINA*
337. REM
338. REM          *CARGA RUTINA*
339. REM
340. REM          *CARGA RUTINA*
341. REM
342. REM          *CARGA RUTINA*
343. REM
344. REM          *CARGA RUTINA*
345. REM
346. REM          *CARGA RUTINA*
347. REM
348. REM          *CARGA RUTINA*
349. REM
350. REM          *CARGA RUTINA*
351. REM
352. REM          *CARGA RUTINA*
353. REM
354. REM          *CARGA RUTINA*
355. REM
356. REM          *CARGA RUTINA*
357. REM
358. REM          *CARGA RUTINA*
359. REM
360. REM          *CARGA RUTINA*
361. REM
362. REM          *CARGA RUTINA*
363. REM
364. REM          *CARGA RUTINA*
365. REM
366. REM          *CARGA RUTINA*
367. REM
368. REM          *CARGA RUTINA*
369. REM
370. REM          *CARGA RUTINA*
371. REM
372. REM          *CARGA RUTINA*
373. REM
374. REM          *CARGA RUTINA*
375. REM
376. REM          *CARGA RUTINA*
377. REM
378. REM          *CARGA RUTINA*
379. REM
380. REM          *CARGA RUTINA*
381. REM
382. REM          *CARGA RUTINA*
383. REM
384. REM          *CARGA RUTINA*
385. REM
386. REM          *CARGA RUTINA*
387. REM
388. REM          *CARGA RUTINA*
389. REM
390. REM          *CARGA RUTINA*
391. REM
392. REM          *CARGA RUTINA*
393. REM
394. REM          *CARGA RUTINA*
395. REM
396. REM          *CARGA RUTINA*
397. REM
398. REM          *CARGA RUTINA*
399. REM
400. REM          *CARGA RUTINA*
401. REM
402. REM          *CARGA RUTINA*
403. REM
404. REM          *CARGA RUTINA*
405. REM
406. REM          *CARGA RUTINA*
407. REM
408. REM          *CARGA RUTINA*
409. REM
410. REM          *CARGA RUTINA*
411. REM
412. REM          *CARGA RUTINA*
413. REM
414. REM          *CARGA RUTINA*
415. REM
416. REM          *CARGA RUTINA*
417. REM
418. REM          *CARGA RUTINA*
419. REM
420. REM          *CARGA RUTINA*
421. REM
422. REM          *CARGA RUTINA*
423. REM
424. REM          *CARGA RUTINA*
425. REM
426. REM          *CARGA RUTINA*
427. REM
428. REM          *CARGA RUTINA*
429. REM
430. REM          *CARGA RUTINA*
431. REM
432. REM          *CARGA RUTINA*
433. REM
434. REM          *CARGA RUTINA*
435. REM
436. REM          *CARGA RUTINA*
437. REM
438. REM          *CARGA RUTINA*
439. REM
440. REM          *CARGA RUTINA*
441. REM
442. REM          *CARGA RUTINA*
443. REM
444. REM          *CARGA RUTINA*
445. REM
446. REM          *CARGA RUTINA*
447. REM
448. REM          *CARGA RUTINA*
449. REM
450. REM          *CARGA RUTINA*
451. REM
452. REM          *CARGA RUTINA*
453. REM
454. REM          *CARGA RUTINA*
455. REM
456. REM          *CARGA RUTINA*
457. REM
458. REM          *CARGA RUTINA*
459. REM
460. REM          *CARGA RUTINA*
461. REM
462. REM          *CARGA RUTINA*
463. REM
464. REM          *CARGA RUTINA*
465. REM
466. REM          *CARGA RUTINA*
467. REM
468. REM          *CARGA RUTINA*
469. REM
470. REM          *CARGA RUTINA*
471. REM
472. REM          *CARGA RUTINA*
473. REM
474. REM          *CARGA RUTINA*
475. REM
476. REM          *CARGA RUTINA*
477. REM
478. REM          *CARGA RUTINA*
479. REM
480. REM          *CARGA RUTINA*
481. REM
482. REM          *CARGA RUTINA*
483. REM
484. REM          *CARGA RUTINA*
485. REM
486. REM          *CARGA RUTINA*
487. REM
488. REM          *CARGA RUTINA*
489. REM
490. REM          *CARGA RUTINA*
491. REM
492. REM          *CARGA RUTINA*
493. REM
494. REM          *CARGA RUTINA*
495. REM
496. REM          *CARGA RUTINA*
497. REM
498. REM          *CARGA RUTINA*
499. REM
500. REM          *CARGA RUTINA*
501. REM
502. REM          *CARGA RUTINA*
503. REM
504. REM          *CARGA RUTINA*
505. REM
506. REM          *CARGA RUTINA*
507. REM
508. REM          *CARGA RUTINA*
509. REM
510. REM          *CARGA RUTINA*
511. REM
512. REM          *CARGA RUTINA*
513. REM
514. REM          *CARGA RUTINA*
515. REM
516. REM          *CARGA RUTINA*
517. REM
518. REM          *CARGA RUTINA*
519. REM
520. REM          *CARGA RUTINA*
521. REM
522. REM          *CARGA RUTINA*
523. REM
524. REM          *CARGA RUTINA*
525. REM
526. REM          *CARGA RUTINA*
527. REM
528. REM          *CARGA RUTINA*
529. REM
530. REM          *CARGA RUTINA*
531. REM
532. REM          *CARGA RUTINA*
533. REM
534. REM          *CARGA RUTINA*
535. REM
536. REM          *CARGA RUTINA*
537. REM
538. REM          *CARGA RUTINA*
539. REM
540. REM          *CARGA RUTINA*
541. REM
542. REM          *CARGA RUTINA*
543. REM
544. REM          *CARGA RUTINA*
545. REM
546. REM          *CARGA RUTINA*
547. REM
548. REM          *CARGA RUTINA*
549. REM
550. REM          *CARGA RUTINA*
551. REM
552. REM          *CARGA RUTINA*
553. REM
554. REM          *CARGA RUTINA*
555. REM
556. REM          *CARGA RUTINA*
557. REM
558. REM          *CARGA RUTINA*
559. REM
560. REM          *CARGA RUTINA*
561. REM
562. REM          *CARGA RUTINA*
563. REM
564. REM          *CARGA RUTINA*
565. REM
566. REM          *CARGA RUTINA*
567. REM
568. REM          *CARGA RUTINA*
569. REM
570. REM          *CARGA RUTINA*
571. REM
572. REM          *CARGA RUTINA*
573. REM
574. REM          *CARGA RUTINA*
575. REM
576. REM          *CARGA RUTINA*
577. REM
578. REM          *CARGA RUTINA*
579. REM
580. REM          *CARGA RUTINA*
581. REM
582. REM          *CARGA RUTINA*
583. REM
584. REM          *CARGA RUTINA*
585. REM
586. REM          *CARGA RUTINA*
587. REM
588. REM          *CARGA RUTINA*
589. REM
590. REM          *CARGA RUTINA*
591. REM
592. REM          *CARGA RUTINA*
593. REM
594. REM          *CARGA RUTINA*
595. REM
596. REM          *CARGA RUTINA*
597. REM
598. REM          *CARGA RUTINA*
599. REM
600. REM          *CARGA RUTINA*
601. REM
602. REM          *CARGA RUTINA*
603. REM
604. REM          *CARGA RUTINA*
605. REM
606. REM          *CARGA RUTINA*
607. REM
608. REM          *CARGA RUTINA*
609. REM
610. REM          *CARGA RUTINA*
611. REM
612. REM          *CARGA RUTINA*
613. REM
614. REM          *CARGA RUTINA*
615. REM
616. REM          *CARGA RUTINA*
617. REM
618. REM          *CARGA RUTINA*
619. REM
620. REM          *CARGA RUTINA*
621. REM
622. REM          *CARGA RUTINA*
623. REM
624. REM          *CARGA RUTINA*
625. REM
626. REM          *CARGA RUTINA*
627. REM
628. REM          *CARGA RUTINA*
629. REM
630. REM          *CARGA RUTINA*
631. REM
632. REM          *CARGA RUTINA*
633. REM
634. REM          *CARGA RUTINA*
635. REM
636. REM          *CARGA RUTINA*
637. REM
638. REM          *CARGA RUTINA*
639. REM
640. REM          *CARGA RUTINA*
641. REM
642. REM          *CARGA RUTINA*
643. REM
644. REM          *CARGA RUTINA*
645. REM
646. REM          *CARGA RUTINA*
647. REM
648. REM          *CARGA RUTINA*
649. REM
650. REM          *CARGA RUTINA*
651. REM
652. REM          *CARGA RUTINA*
653. REM
654. REM          *CARGA RUTINA*
655. REM
656. REM          *CARGA RUTINA*
657. REM
658. REM          *CARGA RUTINA*
659. REM
660. REM          *CARGA RUTINA*
661. REM
662. REM          *CARGA RUTINA*
663. REM
664. REM          *CARGA RUTINA*
665. REM
666. REM          *CARGA RUTINA*
667. REM
668. REM          *CARGA RUTINA*
669. REM
670. REM          *CARGA RUTINA*
671. REM
672. REM          *CARGA RUTINA*
673. REM
674. REM          *CARGA RUTINA*
675. REM
676. REM          *CARGA RUTINA*
677. REM
678. REM          *CARGA RUTINA*
679. REM
680. REM          *CARGA RUTINA*
681. REM
682. REM          *CARGA RUTINA*
683. REM
684. REM          *CARGA RUTINA*
685. REM
686. REM          *CARGA RUTINA*
687. REM
688. REM          *CARGA RUTINA*
689. REM
690. REM          *CARGA RUTINA*
691. REM
692. REM          *CARGA RUTINA*
693. REM
694. REM          *CARGA RUTINA*
695. REM
696. REM          *CARGA RUTINA*
697. REM
698. REM          *CARGA RUTINA*
699. REM
700. REM          *CARGA RUTINA*
701. REM
702. REM          *CARGA RUTINA*
703. REM
704. REM          *CARGA RUTINA*
705. REM
706. REM          *CARGA RUTINA*
707. REM
708. REM          *CARGA RUTINA*
709. REM
710. REM          *CARGA RUTINA*
711. REM
712. REM          *CARGA RUTINA*
713. REM
714. REM          *CARGA RUTINA*
715. REM
716. REM          *CARGA RUTINA*
717. REM
718. REM          *CARGA RUTINA*
719. REM
720. REM          *CARGA RUTINA*
721. REM
722. REM          *CARGA RUTINA*
723. REM
724. REM          *CARGA RUTINA*
725. REM
726. REM          *CARGA RUTINA*
727. REM
728. REM          *CARGA RUTINA*
729. REM
730. REM          *CARGA RUTINA*
731. REM
732. REM          *CARGA RUTINA*
733. REM
734. REM          *CARGA RUTINA*
735. REM
736. REM          *CARGA RUTINA*
737. REM
738. REM          *CARGA RUTINA*
739. REM
740. REM          *CARGA RUTINA*
741. REM
742. REM          *CARGA RUTINA*
743. REM
744. REM          *CARGA RUTINA*
745. REM
746. REM          *CARGA RUTINA*
747. REM
748. REM          *CARGA RUTINA*
749. REM
750. REM          *CARGA RUTINA*
751. REM
752. REM          *CARGA RUTINA*
753. REM
754. REM          *CARGA RUTINA*
755. REM
756. REM          *CARGA RUTINA*
757. REM
758. REM          *CARGA RUTINA*
759. REM
760. REM          *CARGA RUTINA*
761. REM
762. REM          *CARGA RUTINA*
763. REM
764. REM          *CARGA RUTINA*
765. REM
766. REM          *CARGA RUTINA*
767. REM
768. REM          *CARGA RUTINA*
769. REM
770. REM          *CARGA RUTINA*
771. REM
772. REM          *CARGA RUTINA*
773. REM
774. REM          *CARGA RUTINA*
775. REM
776. REM          *CARGA RUTINA*
777. REM
778. REM          *CARGA RUTINA*
779. REM
780. REM          *CARGA RUTINA*
781. REM
782. REM          *CARGA RUTINA*
783. REM
784. REM          *CARGA RUTINA*
785. REM
786. REM          *CARGA RUTINA*
787. REM
788. REM          *CARGA RUTINA*
789. REM
790. REM          *CARGA RUTINA*
791. REM
792. REM          *CARGA RUTINA*
793. REM
794. REM          *CARGA RUTINA*
795. REM
796. REM          *CARGA RUTINA*
797. REM
798. REM          *CARGA RUTINA*
799. REM
800. REM          *CARGA RUTINA*
801. REM
802. REM          *CARGA RUTINA*
803. REM
804. REM          *CARGA RUTINA*
805. REM
806. REM          *CARGA RUTINA*
807. REM
808. REM          *CARGA RUTINA*
809. REM
810. REM          *CARGA RUTINA*
811. REM
812. REM          *CARGA RUTINA*
813. REM
814. REM          *CARGA RUTINA*
815. REM
816. REM          *CARGA RUTINA*
817. REM
818. REM          *CARGA RUTINA*
819. REM
820. REM          *CARGA RUTINA*
821. REM
822. REM          *CARGA RUTINA*
823. REM
824. REM          *CARGA RUTINA*
825. REM
826. REM          *CARGA RUTINA*
827. REM
828. REM          *CARGA RUTINA*
829. REM
830. REM          *CARGA RUTINA*
831. REM
832. REM          *CARGA RUTINA*
833. REM
834. REM          *CARGA RUTINA*
835. REM
836. REM          *CARGA RUTINA*
837. REM
838. REM          *CARGA RUTINA*
839. REM
840. REM          *CARGA RUTINA*
841. REM
842. REM          *CARGA RUTINA*
843. REM
844. REM          *CARGA RUTINA*
845. REM
846. REM          *CARGA RUTINA*
847. REM
848. REM          *CARGA RUTINA*
849. REM
850. REM          *CARGA RUTINA*
851. REM
852. REM          *CARGA RUTINA*
853. REM
854. REM          *CARGA RUTINA*
855. REM
856. REM          *CARGA RUTINA*
857. REM
858. REM          *CARGA RUTINA*
859. REM
860. REM          *CARGA RUTINA*
861. REM
862. REM          *CARGA RUTINA*
863. REM
864. REM          *CARGA RUTINA*
865. REM
866. REM          *CARGA RUTINA*
867. REM
868. REM          *CARGA RUTINA*
869. REM
870. REM          *CARGA RUTINA*
871. REM
872. REM          *CARGA RUTINA*
873. REM
874. REM          *CARGA RUTINA*
875. REM
876. REM          *CARGA RUTINA*
877. REM
878. REM          *CARGA RUTINA*
879. REM
880. REM          *CARGA RUTINA*
881. REM
882. REM          *CARGA RUTINA*
883. REM
884. REM          *CARGA RUTINA*
885. REM
886. REM          *CARGA RUTINA*
887. REM
888. REM          *CARGA RUTINA*
889. REM
890. REM          *CARGA RUTINA*
891. REM
892. REM          *CARGA RUTINA*
893. REM
894. REM          *CARGA RUTINA*
895. REM
896. REM          *CARGA RUTINA*
897. REM
898. REM          *CARGA RUTINA*
899. REM
900. REM          *CARGA RUTINA*
901. REM
902. REM          *CARGA RUTINA*
903. REM
904. REM          *CARGA RUTINA*
905. REM
906. REM          *CARGA RUTINA*
907. REM
908. REM          *CARGA RUTINA*
909. REM
910. REM          *CARGA RUTINA*
911. REM
912. REM          *CARGA RUTINA*
913. REM
914. REM          *CARGA RUTINA*
915. REM
916. REM          *CARGA RUTINA*
917. REM
918. REM          *CARGA RUTINA*
919. REM
920. REM          *CARGA RUTINA*
921. REM
922. REM          *CARGA RUTINA*
923. REM
924. REM          *CARGA RUTINA*
925. REM
926. REM          *CARGA RUTINA*
927. REM
928. REM          *CARGA RUTINA*
929. REM
930. REM          *CARGA RUTINA*
931. REM
932. REM          *CARGA RUTINA*
933. REM
934. REM          *CARGA RUTINA*
935. REM
936. REM          *CARGA RUTINA*
937. REM
938. REM          *CARGA RUTINA*
939. REM
940. REM          *CARGA RUTINA*
941. REM
942. REM          *CARGA RUTINA*
943. REM
944. REM          *CARGA RUTINA*
945. REM
946. REM          *CARGA RUTINA*
947. REM
948. REM          *CARGA RUTINA*
949. REM
950. REM          *CARGA RUTINA*
951. REM
952. REM          *CARGA RUTINA*
953. REM
954. REM          *CARGA RUTINA*
955. REM
956. REM          *CARGA RUTINA*
957. REM
958. REM          *CARGA RUTINA*
959. REM
960. REM          *CARGA RUTINA*
961. REM
962. REM          *CARGA RUTINA*
963. REM
964. REM          *CARGA RUTINA*
965. REM
966. REM          *CARGA RUTINA*
967. REM
968. REM          *CARGA RUTINA*
969. REM
970. REM          *CARGA RUTINA*
971. REM
972. REM          *CARGA RUTINA*
973. REM
974. REM          *CARGA RUTINA*
975. REM
976. REM          *CARGA RUTINA*
977. REM
978. REM          *CARGA RUTINA*
979. REM
980. REM          *CARGA RUTINA*
981. REM
982. REM          *CARGA RUTINA*
983. REM
984. REM          *CARGA RUTINA*
985. REM
986. REM          *CARGA RUTINA*
987. REM
988. REM          *CARGA RUTINA*
989. REM
990. REM          *CARGA RUTINA*
991. REM
992. REM          *CARGA RUTINA*
993. REM
994. REM          *CARGA RUTINA*
995. REM
996. REM          *CARGA RUTINA*
997. REM
998. REM          *CARGA RUTINA*
999. REM
1000. REM          *CARGA RUTINA*
1001. REM
1002. REM          *CARGA RUTINA*
1003. REM
1004. REM          *CARGA RUTINA*
1005. REM
1006. REM          *CARGA RUTINA*
1007. REM
1008. REM          *CARGA RUTINA*
1009. REM
1010. REM          *CARGA RUTINA*
1011. REM
1012. REM          *CARGA RUTINA*
1013. REM
1014. REM          *CARGA RUTINA*
1015. REM
1016. REM          *CARGA RUTINA*
1017. REM
1018. REM          *CARGA RUTINA*
1019. REM
1020. REM          *CARGA RUTINA*
1021. REM
1022. REM          *CARGA RUTINA*
1023. REM
1024. REM          *CARGA RUTINA*
1025. REM
1026. REM          *CARGA RUTINA*
1027. REM
1028. REM          *CARGA RUTINA*
1029. REM
1030. REM          *CARGA RUTINA*
1031. REM
1032. REM          *CARGA RUTINA*
1033. REM
1034. REM          *CARGA RUTINA*
1035. REM
1036. REM          *CARGA RUTINA*
1037. REM
1038. REM          *CARGA RUTINA*
1039. REM
1040. REM          *CARGA RUTINA*
1041. REM
1042. REM          *CARGA RUTINA*
1043. REM
1044. REM          *CARGA RUTINA*
1045. REM
1046. REM          *CARGA RUTINA*
1047. REM
1048. REM          *CARGA RUTINA*
1049. REM
1050. REM          *CARGA RUTINA*
1051. REM
1052. REM          *CARGA RUTINA*
1053. REM
1054. REM          *CARGA RUTINA*
1055. REM
1056. REM          *CARGA RUTINA*
1057. REM
1058. REM         
```

UTILIZANDO EL PORT DEL USUARIO



Trata acerca del manejo del Port de salida del usuario con objeto de extender las aplicaciones y controles del VIC-20 más allá del propio equipo, controlando y recibiendo señales del exterior.

El tema no está desarrollado exhaustivamente, mas bien intenta iniciar al usuario, para que con la imaginación que normalmente se posee, pueda desarrollarse en este campo.

El VIC-20 dispone de un Port libre para el usuario y accesible directamente por el conector trasero de entradas/salidas usuario. Este Port que es el Port B del circuito integrado interfaz 6522) va contraseñado como PB0 a PB7, según el esquema del conector, figura 1.



Ernesto Sánchez Gutiérrez
C/Mayor, nº 11
41002 DOZO
40400000

En primer lugar lo que hay que hacer es preparar el Port B para salida de datos, lo que se consigue mediante la instrucción:

POKE 37136,A

donde A es un número que puede variar entre 0 y 255 según las salidas (PB0 a PB7) que se pretenda utilizar. Por ejemplo, si se desea sacar señales por el PB0, PB1 y PB6, el valor de A debería ser:

$$1 + 8 + 64 = 73 = A$$

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

ENTRADA/SALIDA USUARIO



Contacto nº	TIPO	NOTA	Contacto nº	TIPO	NOTA
1	Tierra		A	10 HRR A	
2	+5V	100mA M.A.N.	B	CR1	
3	RS-171		C	PB0	
4	Bus 0		D	PB1	
5	Bus 1		E	PB2	
6	Bus 2		F	PB3	
7	Capacitor		H	PB4	
8	Interruptor casette		I	PB5	
9	Entrada Serial, ATN		K	PB6	
10	+9V	100mA M.A.N.	L	PB7	
11	Tierra		M	CR2	
12	Tierra		N	10 HRR A	

Figura 1

entonces al hacer POKE 37136,73 habremos "abrir" las salidas PB0, PB1 y PB2. Estas salidas estarán ahora a nivel bajo (0 voltios). Si queremos obtener por ellas un nivel alto de tensión debemos hacer POKE 37136,H

viendo H el número similar al anterior A de tal forma que si B = 73 al hacer POKE 37136,73 las salidas PB0, PB1 y PB2 pasarán al nivel alto de tensión.

Si hacemos POKE 37136,I sólo pasará la PB0 a nivel alto. Igualmente si escribimos POKE 37136,J sólo pasará la PB1 a nivel alto y si hacemos POKE 37136,K sólo lo hará la PB6.

Un ejemplo práctico que aclarará aun mas las cosas es el que figura al principio de la siguiente página.

Cubrir el siguiente montaje



Se escribimos el siguiente programa...

```
10 PRINT "Ciclo"
20 PRINT "EMPLZAMOS"
30 POKE 37136,73: REM ABRIR
  EL PORT B DEL USUARIO
  PARA SALIDA
40 GOSUB 500/
50 GOTO 60
60 POKE 37136,8: REM ENCEN-
  DIZ R 11:11:1 (PB0)
70 GOSUB 500/
80 POKE 37136,64: REM ENCEN-
  DIZ R 11:11:1 (PB0)
90 GOSUB 500/
100 POKE 37136,73: REM ENCEN-
  DIZ TODOS LOS LED
```

```
110 GOSUB 500
120 GOTO 40
500 REM TIEMPO DE ENCENDIDO
510 FOR I = 1 TO 500
520 NEXT I
530 RETURN
```

y lo hacemos rodar (RUN (RETURN)), conseguiremos encender paulatinamente los tres LEDs, de uno en uno y al final del ciclo los tres al mismo tiempo. Está claro? Pues imagínate las posibilidades que tienes.

Ox voy a comentar otro ejemplo más práctico del manejo del Port del usuario, controlaremos los semáforos de un supuesto cruce de calles como los representados en la figura 2.

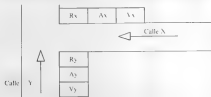


Figura 2

Si reanimamos el pequeño montaje de la figura 3 y lo conectamos a los

puntos del conector de entradas/salidas usuario tal como se indica

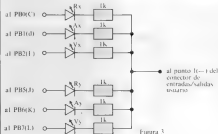


Figura 3

...y escribimos el siguiente programa de control...

```
10 PRINT "CICLO EMPLZAMOS"
20 POKE 37136,231: REM ABRIR
  EL PORT B PARA SALIDA
  DE DATOS
300 PRINT "ROJO X", "VERDE Y"
400 PRINT "ROJO X", "AMBAR Y"
500 PRINT "VERDE X", "ROJO Y"
600 PRINT "VERDE X", "ROJO Y"
700 PRINT "VERDE X", "ROJO Y"
800 PRINT "VERDE X", "ROJO Y"
900 PRINT "VERDE X", "ROJO Y"
1000 PRINT "VERDE X", "ROJO Y"
1100 PRINT "VERDE X", "ROJO Y"
1200 PRINT "VERDE X", "ROJO Y"
1300 PRINT "VERDE X", "ROJO Y"
1400 PRINT "VERDE X", "ROJO Y"
1500 PRINT "VERDE X", "ROJO Y"
1600 PRINT "VERDE X", "ROJO Y"
1700 PRINT "VERDE X", "ROJO Y"
1800 PRINT "VERDE X", "ROJO Y"
1900 PRINT "VERDE X", "ROJO Y"
2000 PRINT "VERDE X", "ROJO Y"
2100 PRINT "VERDE X", "ROJO Y"
2200 PRINT "VERDE X", "ROJO Y"
2300 PRINT "VERDE X", "ROJO Y"
2400 PRINT "VERDE X", "ROJO Y"
2500 PRINT "VERDE X", "ROJO Y"
2600 PRINT "VERDE X", "ROJO Y"
2700 PRINT "VERDE X", "ROJO Y"
2800 PRINT "VERDE X", "ROJO Y"
2900 PRINT "VERDE X", "ROJO Y"
3000 PRINT "VERDE X", "ROJO Y"
3100 PRINT "VERDE X", "ROJO Y"
3200 PRINT "VERDE X", "ROJO Y"
3300 PRINT "VERDE X", "ROJO Y"
3400 PRINT "VERDE X", "ROJO Y"
3500 PRINT "VERDE X", "ROJO Y"
3600 PRINT "VERDE X", "ROJO Y"
3700 PRINT "VERDE X", "ROJO Y"
3800 PRINT "VERDE X", "ROJO Y"
3900 PRINT "VERDE X", "ROJO Y"
4000 PRINT "VERDE X", "ROJO Y"
4100 PRINT "VERDE X", "ROJO Y"
4200 PRINT "VERDE X", "ROJO Y"
4300 PRINT "VERDE X", "ROJO Y"
4400 PRINT "VERDE X", "ROJO Y"
4500 PRINT "VERDE X", "ROJO Y"
4600 PRINT "VERDE X", "ROJO Y"
4700 PRINT "VERDE X", "ROJO Y"
4800 PRINT "VERDE X", "ROJO Y"
4900 PRINT "VERDE X", "ROJO Y"
5000 PRINT "VERDE X", "ROJO Y"
5100 PRINT "VERDE X", "ROJO Y"
5200 PRINT "VERDE X", "ROJO Y"
5300 PRINT "VERDE X", "ROJO Y"
5400 PRINT "VERDE X", "ROJO Y"
5500 PRINT "VERDE X", "ROJO Y"
5600 PRINT "VERDE X", "ROJO Y"
5700 PRINT "VERDE X", "ROJO Y"
5800 PRINT "VERDE X", "ROJO Y"
5900 PRINT "VERDE X", "ROJO Y"
6000 PRINT "VERDE X", "ROJO Y"
6100 PRINT "VERDE X", "ROJO Y"
6200 PRINT "VERDE X", "ROJO Y"
6300 PRINT "VERDE X", "ROJO Y"
6400 PRINT "VERDE X", "ROJO Y"
6500 PRINT "VERDE X", "ROJO Y"
6600 PRINT "VERDE X", "ROJO Y"
6700 PRINT "VERDE X", "ROJO Y"
6800 PRINT "VERDE X", "ROJO Y"
6900 PRINT "VERDE X", "ROJO Y"
7000 PRINT "VERDE X", "ROJO Y"
7100 PRINT "VERDE X", "ROJO Y"
7200 PRINT "VERDE X", "ROJO Y"
7300 PRINT "VERDE X", "ROJO Y"
7400 PRINT "VERDE X", "ROJO Y"
7500 PRINT "VERDE X", "ROJO Y"
7600 PRINT "VERDE X", "ROJO Y"
7700 PRINT "VERDE X", "ROJO Y"
7800 PRINT "VERDE X", "ROJO Y"
7900 PRINT "VERDE X", "ROJO Y"
8000 PRINT "VERDE X", "ROJO Y"
8100 PRINT "VERDE X", "ROJO Y"
8200 PRINT "VERDE X", "ROJO Y"
8300 PRINT "VERDE X", "ROJO Y"
8400 PRINT "VERDE X", "ROJO Y"
8500 PRINT "VERDE X", "ROJO Y"
8600 PRINT "VERDE X", "ROJO Y"
8700 PRINT "VERDE X", "ROJO Y"
8800 PRINT "VERDE X", "ROJO Y"
8900 PRINT "VERDE X", "ROJO Y"
9000 PRINT "VERDE X", "ROJO Y"
9100 PRINT "VERDE X", "ROJO Y"
9200 PRINT "VERDE X", "ROJO Y"
9300 PRINT "VERDE X", "ROJO Y"
9400 PRINT "VERDE X", "ROJO Y"
9500 PRINT "VERDE X", "ROJO Y"
9600 PRINT "VERDE X", "ROJO Y"
9700 PRINT "VERDE X", "ROJO Y"
9800 PRINT "VERDE X", "ROJO Y"
9900 PRINT "VERDE X", "ROJO Y"
10000 PRINT "VERDE X", "ROJO Y"
```

Si pensamos que en el transcurso de cada ciclo se deben producir interrupciones inesperadas, podemos añadir alguna subrutina que por ejemplo ponga en rojo o en ámbar momentáneamente los semáforos. Estas subrutinas podrían ser así:

```
500 REM INTERRUPTOR DEL
  CICLO NORMAL
510 GOTO 520
520 IF AS = "1" THEN GOTO 530
530 IF AS = "0" THEN GOTO 540
540 IF AS = "1" THEN GOTO 550
550 RETURN
600 PRINT "ROJO X", "ROJO Y"
  REM SEMAFORO A ROJO
  HASTA VOLVER A SITUACION
  NORMAL
610 POKE 37136,11
620 GOSUB 500
630 GOTO 610
700 PRINT "AMBAR X", "AMBAR Y"
  REM SEMAFORO A AMBAR
  INTERMITENTES HASTA VOLVER
  AL CICLO NORMAL
710 POKE 37136,66
720 FOR I = 1 TO 500
730 NEXT I
740 GOSUB 500
750 POKE 37136,0
760 FOR I = 1 TO 500/
770 NEXT I
780 GOSUB 500
790 GOTO 710
```

Habría que incluir en el programa principal las llamadas a las subrutinas de interrupción, escribiendo:

```
115 GOSUB 500
215 GOSUB 500
315 GOSUB 500
415 GOSUB 500
```



Ave si durante un ciclo normal se desea una interrupción, se podría lograr pulsando "1" o "0" a los contadores se pondrían en "0" o en "ambos" intermitentemente respectivamente, no sabiendo de esa situación hasta que se pulse la tecla "C" retornando entonces al ciclo normal.

Igual que este ejemplo se podría haber programado una secuencia de luces, etc.

Evidentemente en caso de querer controlar algo de más potencia que unos simples LEDs, deberíamos conectar un interfaz adecuado a la potencia controlada, de tal forma que no se Jare el Port B utilizado del VIC-20.

Hasta aquí hemos utilizado las posibilidades del Port, para circuitos sencillos al exterior, pero por programa EIO o por interrupciones controladas

por el teclado. Pero podríamos pensar en alguna aplicación en la que la señal de mundo provenga del exterior del equipo, como si fuera un terminatio presostato final de carrera, etc.

En ese caso podríamos utilizar el Port A del mismo circuito integrado interfaz 8522. O sea lo que controlamos como $IOY0$, $IOY1$, $IOY2$ y $IOY3$ PEN, que son los más directamente accesibles.

El POKI que vamos a utilizar para controlar si alguna de estas entradas ha pasado a 0 colamos el 37157, de tal manera que podemos efectuar las condiciones:

```
1000 PEPA, PE1K (37157)
1010 IF (PEPA AND 4) = 0 THEN
5000
1020 IF (PEPA AND 8) = 0 THEN
6000
1030 IF (PEPA AND 16) = 0 THEN
7000
1040 IF (PEPA AND 32) = 0 THEN
8000
```

5000 REM JOY 0 ACTIVADO

6000 REM JOY 1 ACTIVADO

7000 REM JOY 2 ACTIVADO

8000 REM LITE-PEN-ACTIVADO

Estando en 5000, 6000, 7000 y 8000 los programas de control de los juegos externos que podemos generar a partir del PORT B como ya sabemos hacerlo.

Con estas breves nociones quiero abrir la puerta para que podamos ampliar las posibilidades de nuestro VIC-20.

Para todos los usuarios de la informática personal

MS

MicroSistemas

256 pag.

Edición mensual
de Computerworld/España

Para todos los usuarios de la informática personal

MS
MicroSistemas



**TODO
SOBRE APPLE EN
ESPAÑA**

Guía de Apple España

**Ejemplar atrasado
250 pesetas**

**Colección completa
(6 ejemplares)
1.250 pesetas**

Para todos los usuarios de la informática personal

MS
MicroSistemas



**EL COLEGIO
Y LA INFORMÁTICA**



Hemos contactado con los espíritus más poderosos del mundo de la informática para que nos trajeran esta selección de consejos útiles, indicaciones y métodos. Estas maravillas de la magia les van a encantar con unos métodos fantásticos para que Ud. aproveche al máximo su sistema.

Magia



La MAGIA son trucos, la MAGIA es divertida.

La MAGIA es hacer lo que nadie se ha atrevido.

La MAGIA es una columna mensual llena de consejos, trucos, de esto y aquello del mundo del software, hardware y aplicaciones.

Cada mes, MAGIA les trae trucos breves y útiles de informática procedentes de todo el mundo - trucos descubiertos por los demás que hacen que la informática sea más fácil, más divertida o más avanzada.

MAGIA habla de ideas sencillas, programas de una sola línea, subrutinas útiles, hechos de informática poco conocidos y otras

cosas de interés. Buscamos material nuevo o renovado que resulte ser de valor actual para usuarios de equipos Commodore y que puede utilizarse con un mínimo de tiempo, esfuerzo o conocimientos técnicos.

MAGIA resulta ser la fuente más completa de información para la informática práctica, además de ser un foro internacional para compartir trucos con otros aficionados. Envíe sus trucos a:

COMMODORE WORLD
Pedro Magarza, 4
Madrid-18

La revista "Commodore World" sorteará orígenes de software en julio y diciembre entre todas las contribuciones publicadas.

El truco especial "de una sola línea" de este mes es bastante antiguo —es de 1978—; cuando un Commodore PET de 8K no valía lo que vale ahora y era imposible conseguir ningún tipo de documentación para leer. No había libros, y las únicas revistas eran hojas informativas producidas por usuarios no profesionales.

The PET Gazette era una de estas hojas, y aquí ofrecemos uno de sus trucos más antiguos, llamado "BURROW".

IAS="arriba"abajo"quienquiera"derecha"PRINTMID\$(ASC(RN\$(3*4+1),1)*"izquierda")::FORI=1TO30NEXT PRINTY"iva on"espacio"izquierda")::GOTO1

Cabe en una línea de 40 columnas, y se hace cada vez más divertido.

Nos gustaría ver sus programas de una sola línea, y queremos publicar por lo menos uno cada mes. Los programas pueden ser divertidos, de humor, útiles o no, con tal de que quepan en 40 columnas o menos. ¿Y usted, qué nos puede ofrecer?

Montse Ferrán

* * *

Muchas sentencias caben en una línea de programa mediante el uso de las abreviaturas de las palabras clave de Basic incluidas en el apéndice del manual del usuario. Cuando la línea se produce en forma de listado, las palabras clave se imprimen sin abreviar, con el resultado de que la línea del programa podría ocupar más que el número normal de líneas en la pantalla.

Esto no es problema, pero al intentar editar esta línea larga, el ordenador la reducirá para que se ajuste a la longitud normal para una línea de programa. Así que se pueden utilizar las abreviaturas para poder incluir las sentencias en una línea, pero tenga mucho cuidado al editarla más adelante.

Pittsburgh Commodore Group Newsletter

* * *

Todos los 16 colores del Commodore 64 pueden ser llamados desde el teclado, pero sólo ocho de ellos pueden ser reconocidos en las teclas. Si vd. pega un trozo de cinta adhesiva de unas 6 pulgadas encima de las teclas numéricas de 1-8, puede marcar los otros colores en dicha cinta y todo el proceso resulta mucho más fácil.

Los colores se llaman al pulsar la tecla Logo del Commodore junto con una tecla numérica. Empezando por la izquierda los colores son, naranja, marrón, rojo claro, gris oscuro, gris mediano, verde claro, azul claro y gris claro.

L.F.S.

* * *

Puede ocurrir que en algunas televisiones la pantalla del VIC no quede exactamente en el centro, con el resultado de que se pierde parte de la representación. La posición 36864 controla el centro horizontal de la pantalla (normalmente 5), y la posición 36864 controla el centro vertical de la pantalla (normalmente 25). Si vd. cambia estos valores le ayudará a centrar la pantalla de una forma más adecuada.

Westmoreland Commodore Newsletter

La llamada "modalidad de comillas" puede volverle loco cuando el ordenador se encuentra en dicha modalidad y vd. quiere salir de ella. Normalmente se sale de la modalidad de comillas al teclear mas comillas, y luego borrarlas.

Pero hay ocasiones en que esto no funciona, como cuando están llenándose los espacios abiertos mediante la tecla de inserción. Dichos espacios se comportan como si la modalidad de comillas fuese activa, aunque no lo sea, y si se teclea otra comilla no se soluciona nada. Si vd. pulsa cualquier tecla de control del cursor cuando se encuentra con un espacio de inserción, casi siempre se consigue la versión de la modalidad de comillas de dicho control de cursor, que normalmente no es lo que estamos buscando. Si intenta borrarlo, se consigue la T del campo invertido, que obviamente, complica aun más el problema.

¿Cuál es la solución? Pulse la tecla de "shift return", que desplaza el cursor a la línea siguiente sin "introducir" la línea que se está modificando. También borra la modalidad de comillas en los espacios de inserción de forma que el cursor pueda volver a dicha línea para seguir con su trabajo.

Tomás Järegui

* * *

La expresión matemática entre If y Then determina si el resto de una sentencia será ejecutada. Cuando la expresión es falsa, el resto de la línea se salta.

Esta característica puede utilizarse para abaratar el tiempo de ejecución. En vez de usar una sentencia como 100 IF X = 1 AND Y = 2 THEN PRINT Z, resulta mucho más rápido escribir 100 IF X = 1 THEN IF Y = 2 THEN N PRINT Z.

En el primer caso, X = 1 AND Y = 2 tiene que ser evaluado antes de tomar cualquier decisión sobre el salto de línea. En el segundo caso, en cuanto se evalúa X = 1 como falso, todo lo demás se salta. Esto resulta en una ejecución más rápida cuando X = 1 es falso.

Roberto F. Rodríguez

* * *

Aquí viene la forma de cronometrar la ejecución de dos programas semejantes.

```
100 TIS = "000000"
110 FORI = 1 TO 500
120 Se incluye aquí el código que se comprueba
130 Etc.
140 Etc.
180 NEXT
190 PRINT TI
```

Se procesa el programa con una versión del programa, y se anota el valor del TI, que resulta ser el número de segundos que tardó en ejecutarse 500 veces. Luego se sustituye el programa de prueba con la otra versión y se vuelve a procesar el programa. La versión que requiere menos segundos en ejecutarse es la más rápida.

L.F.S.

* * *



Con frecuencia se consigue un número negativo al comprobar FRI(0) en el Commodore 64. (Esto no significa que la memoria disponible es negativa; tiene que ver con la forma en que FRE representa los números.) Para convertir el valor negativo en su forma correcta, se realiza la siguiente: $PRN(FRI(0) + 1000000) \text{ lb}$.

Además, el cero en FRI(0) no es nada mágico, cualquier letra o número puede incluirse aquí. Normalmente resulta más fácil encontrar FRE(9) en el teclado, y da el mismo resultado.

Carmen Echevarría

* * *

Utilice tiras de Scotch Magic Tape (papel de celofán para etiquetar las cassettes, se puede escribir claramente con lápiz y borrarlo fácilmente. Pero tenga cuidado de que los trozos de la goma de borrar no caigan dentro de la cinta o la cassette.

Jordi García

* * *

Al guardar un programa en una cinta, añada el RVS al nombre del programa. De esta forma, al volver a leerlo, imprime el nombre sobre un fondo blanco, facilitando su localización. Simplemente teclee `SAVE "(rsv on) NOMBRE DEL PROGRAMA (rsv off)" (return)`.

The PET Gazette

* * *

Si vd. tiene un programa almacenado en la memoria y quiere ejecutarlo, no hace falta teclear la palabra RUN. Simplemente teclee cualquier letra o letras (números no), y pulse la tecla en "SHIFT" run/stop. El programa se ejecutará.

Moncho Zabaleta

Se vd. ha quitado el precinto de una cinta que la protege para que no pueda grabarse encima, existen dos formas de burlarse de esta protección. Una forma es tapando el agujero con un trozo de papel de celo. Otra es enganchar el Datacassette para que pisme que el agujero se ha tapado.

Abra la funda del Datacassette y localice el alfiler pequeño que se encuentra en el agujero vacío. (Se encuentra muy al fondo, en la extrema izquierda.) Empuje suavemente el alfiler hacia la parte de atrás del aparato y pulse el botón de grabar. Introduzca la cinta y vuelva a pulsar el botón de grabar mientras pulsa el botón de play.

Frank O'Connor

* * *

No hace falta colocar paréntesis alrededor del número después de una sentencia `SYS SYS28, SYS 828` y `SYS(828)` significan lo mismo para el ordenador.

L.F.S.

* * *

Podría resultar problemático introducir gráficos o letras en "SHIFT" en una sentencia `RIM` - salen en el listado de una forma rara, normalmente como palabras clave de Basic. Para que se comporten como es debido, póngalos entre parentesis.

J. Luis G. Diego

* * *

Una forma rápida de saber si un número entero es par o impar es hacerlo un AND con un 1. Si el resultado es cero, el número es par, si el resultado es uno, el número es impar. Para que funcione el truco, el número tiene que encontrarse dentro de -32768 a +32767; si no es así, el resultado será un error de cantidades no válidas.

Illego Fernández

MARKETCLUB

—Servicio gratuito para nuestros lectores particulares. Empresas 300 ptas. por línea.

• **CVI 4.832**, intercambio programas. José María Morales, C/le Sevilla, 5, tel. (93) 883 77 51, de 8 a 3. VILANOVA DEL CAMÍ (Barcelona).

• **ACCUSOFTS VIL-20**, Ampliación de memoria 16k, más varios programas, 13.000. Modelo de expansión para 8 cartuchos, 10.000. Cartucho lenguaje FORTH y manual, 7.000. Los tres cosas sólo por 25.000. Janso, tel. 245 46 56. BARCELONA.

• En Barcelona, clases de información. PLAZAS LIMITADAS. Lenguaje BASIC, Practico con micro-ordenador VIL-20, Prof. J. Martínez de Araya, Informatic, tel. (93) 345 10 00. Seixanta Marea José (mañana) o (93) 345 83 75. Sr. Martínez (fuera de horas de oficina).

• Dinero a credit por 40.000 ptas., VIL-20, con cartucho Super Expander, los dos gratis del Curso, el "pre-quick", y un regalo inesperado, toda comprada en diciembre del 82. Fernando Vianer, calle La Roda, 39, 5º D. Teléfono 33 41 82. ALBAYCETE.

• Tengo un PET 2001/86 y desearía tener el cassette "Memoria para lenguaje Máquina". También desearía contactar con usuarios o clubs de PET si los hay. José Manuel Cámara Mas, calle Castro, 33, bloque II, 3º, puerta 1. ALICANTE.

• Programación de ordenadores personales, organización explotación de ficheros, programas ordenadores auxiliares, para cuestiones empresaria-

les, profesionales, administrativas, científicas. María Mas, Carlos III, 45, teléfono 339 98 29. BARCELONA-28.

• Venta empresa Ventisca C/P-100-53, para VIL-20, Commodore 64 como nueva 43.000 ptas. Regalo muchos programas gráficos al comprador. El pasar horas comiendo y con. Ives, Avaterra, tel. 253 13 48. Santiago Revilla, 12. MADRID-3.

• Venta VIL-20 con expansión 36k, superexpansión, curso programación BASIC (dos dos nombres), guía de referencia del programador y cartucho de juegos. Todo en perfecto estado. Precio: 55.000 ptas. Angel Rialser, teléfono (954) 488 320. NAVILLA.

• Venta VIL-20, ampl. de memoria 36k, RAM, cassette, Libros: Manual VIL-20, Referencia, VIL Revealed y 25 programas muy buenos: "Comoceros", "Blitz", "5 de Escuadras", "Matrices", etc. Por sólo 55.000 ptas. David Badalona, 182, tel. (91) 734 11 93. MADRID-34.

• Tengo programa para confección de documentos, cartas, felicitos, etc. Permiten escribir líneas reales de impresora visualizando por pantalla y avisando al final de cada línea por nombre. Visualización del texto interno modificaciones de línea; insertar líneas, grabación disco o cassette; lectura de datos de disco o cassette; control por pantalla del texto a reemplazar; abriga; espacios, etc.; Impresión normal y doble ancho; pregunta número de copias; salto automático de página en impresora; margen izquierdo en la página. Amador Barger. Pda. Hospital 5-97. Teléfono 834 41 93. Vitoria. BARCELONA.

SOFTWARE PARA EL 700



En los dos artículos anteriores he descrito algunos de los más importantes aspectos del 700 a nivel de características técnicas y sistema operativo. Pero el aspecto más importante en todo ordenador es el software, y el 700 no puede ser una excepción. Cuando una persona (o empresa) decide instalar un ordenador es por alguna de las siguientes causas:

—El trabajo administrativo desborda al departamento de administración.

—Los directivos precisan de una información estadística, que exige gran trabajo de elaboración manual, mientras que un ordenador puede suministrarla rápida y fácilmente.

—Debido al volumen de trabajo del departamento de contabilidad, almacén, estadística o facturación, continuamente se están corrigiendo errores, pensando en qué pasará con los errores que no pueden corregirse porque no se han detectado.

—Se desea efectuar un control más estricto sobre los clientes de la empresa (ingreso, impagados...) para evitar "sorpresas" financieras.

—Actualmente se están contratando los servicios de un centro de cálculo. Pero el coste del mismo, el inconveniente de no tener la información "a mano", y el peligro que implica el que se manejen los datos de la empresa fuera de ella, hacen pensar que es mejor disponer de un ordenador propio.

—Por causas diversas: porque fulano tiene uno, porque un amigo me lo aconseja (generalmente ese amigo es representante de alguna marca)... etc.

Entonces, cuando la empresa envía a una persona para que "busque ordenador", se encuentra con que no

Los capítulos previos fueron publicados con anterioridad en COMMODORE CLUB. Aquellos lectores que los deseen, les rogamos nos lo comuniquen para poder enviarles copia de estos capítulos.

JORDI SASTRE

tiene ningún conocimiento de informática y se da cuenta de que fácilmente puede ser objeto de engaños. Una posible solución es contratar los servicios de una empresa asesora en informática para que le mecanice la empresa. Pero actualmente, el mundo de los ordenadores, y más concretamente el mundo de los microordenadores, está creciendo de tal manera, que nadie puede estar totalmente al día en cuanto a máquinas y programas disponibles en cada momento. Cada día aparecen nuevos paquetes

estándar de contabilidad, facturación o nóminas, siempre mejores que los anteriores. Cada día aparecen en el mercado nuevos ordenadores, en principio mejores que los anteriores (pero no todos). Un día el ordenador vale tanto, pero al día siguiente ha aparecido un modelo mejor o más barato. En resumen, según el día de la semana en que se empiece a "buscar ordenador", se acaba comprando una marca u otra.

Actualmente, las empresas distribuidoras de microordenadores, buscan el mayor número de programas estándar posible. Porque un cliente puede ser engañado si un programa le sirve o no, siempre y cuando ese programa esté hecho y sea como trabaja. Los programas a medida no sirven hasta que están hechos, por lo que el cliente no sabía si el ordenador cumplía sus exigencias hasta que el programa está acabado, después de largas semanas de lucha con el programador, y siempre recordándole las promesas del vendedor.

El programa estándar tiene el inconveniente en el mismo. En la mayoría de los casos (sobre todo en gestión comercial y facturación) cuando una empresa adopta un programa estándar es para adaptarse a él, en lugar de adaptar el ordenador a la empresa, que sería lo más deseable.

Un punto intermedio entre el programa estándar y el programa a medida, es el programa a medida confeccionado muy rápidamente con un buen sistema operativo. ¡Aquí queda llegar!

Más importante que las características externas, más importante que todos los programas llamados estándar que pudiera haber, y más importante que cualquiera de las promesas



que pueda hacer un vendedor, el sistema operativo es quien define en verdad la potencia del ordenador. De acuerdo que un buen sistema operativo debe ir soportado por una buena máquina, pero hay que saber por donde empezar a mirar.

El sistema operativo y, por tanto, el lenguaje o lenguajes de programación que utiliza, intervienen en gran alto grado en la velocidad de programación de los ordenadores. Para hacer un programa podemos estar tres días o tres meses, dependiendo del sistema operativo que estemos empleando.

He aquí la mejor política que puede tener una distribuidora de microordenadores: tener como standard los programas que de verdad pueden ser los contables, proceso de textos, finanzas, etc. y disponer de un buen sistema operativo para que los programas a medida puedan hacerse en el mínimo de tiempo, para que el cliente final sea rápidamente el el ordenador le sirve como, y para bajar los costes de programación, que, finalmente, se están disparando.

Esto es lo que se pretende con la serie 200. Pronto estarán disponibles los programas de proceso de textos y

Calc Result. En quince días hay disponible el MIC/DOS MIC/DOS es un sistema operativo co-residente con el sistema operativo normal del 700, que amplía en unos 50 comandos el Basic de Commodore. Sus objetivos son simplificar la programación, aumentar la velocidad de ejecución, en una palabra, la potencia del ordenador. Pero MIC/DOS es tema para otro artículo. En todo esto estaba pensando MIC/DOS I cuando desarrolló MIC/DOS. MIC/DOS permite que los programas sean más fáciles de

confeccionar, más rápidos en su ejecución, más presentables en su edición de pantalla, más potentes en cálculo matemático, más claros para posteriores modificaciones, más y más.

En breve, el 2000 dispondrá de procesadores de textos Wordcraft, Easy-script, Super-script, programas tipo Visual Basic, Result, bases de datos, etcétera.

En el próximo artículo comentaremos más profundamente lo que es el MIC/DOS.

MEA CULPA:

DEBIDO A PROBLEMAS EN EL MANUAL DEL PROCESO DE TEXTO "EASY SCRIPT" PARA EL COMMODORE 64 NO ESTÁ SUFICIENTEMENTE CLARO COMO SE CARGA EL PROGRAMA EN SU VERSIÓN ACTUAL EN DISCO. DESPUES DE ENCENDER EL EQUIPO PULSAR:

LOAD "", 8, 1*

SEGUIDO DE RETURN Y EL PROCESO DE TEXTO EMPEZARÁ A FUNCIONAR.

B.M.

LA SERIE MICRO-COMMODORE 200

PROGRAMAS STANDARD Y «A MEDIDA» PARA EQUIPOS COMMODORE

VIC-20

- CONTABILIDAD
- GESTIÓN COMERC
- STOCK ALMACENES
- VIDEO CLUB
- ENTRAPUNT
- ETC
-
-
-
-

COMMODORE-64

- CONTABILIDAD
- GESTIÓN COMERC
- CONTROL DE STOCKS
- RECIBOS-ETIQUETAS
- ETC
-
-
-
-
-

SISTEMA 8000

- CONTABILIDAD (10MB)
- GESTIÓN COMER
- 9000 ARTICULOS
- GEST INTEGRADA
- ALMACÉN
- NÓMINAS
- DIRECCIÓN
- AUTOVENTA
- CONTROL SOCIOS
- PRODUCCIÓN

SISTEMA 8000

- FINCAS
- IND. CARNICAS
- EMP LIMPIEZA
- COOPERATIVAS
- TALLERES
- COMPONENTES
- PIENSOS
- COLEG PROFES
- CADENAS MONTAJE
- ETC

Avenida Cesar Augusto, 72 - Telefonos 235682 y 226544
ZARAGOZA 3



Los capítulos previos fueron publicados con anterioridad en **COMMODORE CLUB**. Aquellos lectores que los deseen, les rogamos que nos lo comuniquen para poder enviarles copia de estos capítulos.

COMUNICACION SERIE MEDIANTE EL BUS RS-232C (I)

A juzgar por las preguntas que recibimos en nuestra redacción, existe una clamorosa demanda de información sobre el manejo del interface serie (RS-232) que equipa a los equipos VIC-20 y COMMODORE 64. Como Manel Sans es quien tiene la mayor experiencia en este tema, le hemos pasado la "pelota". El ha preparado para nuestra revista una interesante serie de artículos que derramarán billones de fotones sobre los rincones más oscuros de este sistema estandar de comunicación.

AL SANS

En términos generales existen dos métodos básicos para comunicar un ordenador con sus periféricos o con el mundo exterior. Uno es la transferencia de datos en paralelo y el otro es la comunicación serie o transferencia de datos en serie.

La transferencia de datos en paralelo implica enviar o transmitir 8 bits de datos más o menos separados y transmitir un byte de información en cada transferencia. De todas formas para efectuar una transmisión fiable de información se necesitan algunas señales de control adicionales. Estas señales de control se llaman en inglés "HANDSHAKING" (ya podemos traducirlo por "patates o señales de acuerdo").

Algunos ordenadores utilizan el protocolo de comunicación paralelo IEEE-488 establecido en 1976. Otro standard de manejo de bus muy común es el CENTRONICS PARA THE INTERFACE STANDARD. Este método es el más comúnmente utilizado en muchas impresoras.

El uso de bus para comunicación a largas distancias produce muchos problemas ya que, por cada cable utilizan dife-

rentes corrientes a diferentes velocidades. Esto tiene como efecto que en distancias grandes los bits del dato enviado flotan a su destino a diferentes tiempos.

Aun cuando muchos fabricantes utilizan el sistema IEEE-488 de comunicación, el método más común es la comunicación serie según la norma RS-232C. La comunicación por este sistema se efectúa bit a bit, a una cierta velocidad y por un solo hilo.

La mayor ventaja de este sistema es la posibilidad de transmitir los datos a grandes distancias. Con las modificaciones adicionales este método se puede utilizar para transmitir datos por vía telefónica o por vía satelital.

Esencialmente RS-232C es un nombre para el standard formulado por la ELECTRONIC INDUSTRIES ASSOCIATION (EIA). Este standard describe un tipo de conductores necesarios (PROTOCOL) para el manejo del bus serie entre dos equipos.

El RS-232 ha existido en tres diferentes formas desde su formulación a



principios de 1960. El primer estándar RS-232 fue el RS-232A, totalmente obsoleto en la actualidad, así como la mayoría de los equipos que lo utilizaban. La segunda versión fue el RS-232B, esta también fuera de uso aún cuando existen todavía algunos viejos equipos con este estándar. El actual RS-232C es igual que el antiguo RS-232B excepto en que las señales de datos son invertidas.

Este estándar se ha extendido ya no sólo a su área original, comunicación entre terminales y módems, sino como interconexión entre ordenadores y periféricos como impresoras, plotters, etc.

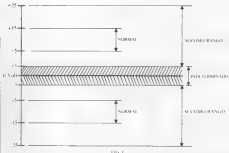
Características generales de las señales

Algunos de los términos usados en comunicación provienen de los ancestrales TTY-TYPE. Los términos para uno y cero lógico, por ejemplo, son MARK y SPACE, respectivamente. La utilización de estos términos en comunicación indicaba la presencia o ausencia de una corriente de 20 miliamperios en el circuito. Algunos teletipos utilizan esta técnica.

El RS-232C ha retenido estos términos, pero usando tensiones para la transmisión de datos en lugar de corrientes. La condición de SPACE ("0") se representa por la presencia de una tensión comprendida entre +3 y +25 voltios y en cambio la condición MARK ("1") se representa por la presencia de una tensión comprendida entre -3 y -25 voltios. Un punto de trabajo entre 5 y 15 voltios es el más usual. Si la tensión positiva o negativa de la transmisión está por debajo de los 3 voltios se produce un estado de indeterminación y el receptor no podrá definir o detectar la condición de esta señal. (Ver figura 1).

Cuando se efectúan transmisiones a través de grandes distancias, la señal en la línea es atenuada ya que como es sabido, el voltaje es inversamente proporcional a la distancia de cable atravesado. El límite práctico en la longitud del cable para este tipo de transmisiones es de alrededor de 300 metros. Para poder transmitir a más largas distancias se debe utilizar buffers de línea especiales que mantengan la señal. Otro método para cubrir largas distancias es el uso de un MODEM (Módulo de Modulación).

Otro problema común en la utilización de grandes longitudes de cable son las interferencias. Estas interferencias son en su mayor parte de naturaleza electromagnética. Algunos ejemplos de fuentes de interferencias electromagnéticas son, luces fluores-



centes, circulación o colocación de los cables cerca de transmisores de radio o televisión, proximidad de instalaciones de RADAR, motores eléctricos, transformadores y muchos más. Para evitar o minimizar estos efectos es aconsejable la utilización de cables apantallados y/o con conductores trenzados. En el caso de que alguno de los conductores no sea utilizado es conveniente unirlos a masa.

Modos de transmisión

En RS-232C son posibles dos modos de transmisión:

- 1) TRANSMISION SINCRONA
- 2) TRANSMISION ASINCRONA

La transmisión SINCRONA mantiene un intervalo de tiempo fijo entre los bits de la transmisión. Los periodos de tiempo son definidos por la estación transmisora mediante el TRANSMIT CLOCK, con el cual la estación receptora sabe exactamente cuando el dato o los bits de dato son válidos.

Los datos SINCRONOS están normalmente organizados en grupos rígidos de caracteres que son llamados colectivamente "PROTOCOLO".

La ventaja en la utilización de la comunicación SINCRONA es que la estación transmisora tiene un control rígido sobre el receptor, y por esta causa el formato del mensaje se conoce por anticipado y los checksums usados para la detección de errores son muy precisos. La principal desventaja en este tipo de transmisión es lo costoso de mantener todos los tipos de mensajes y respuestas. Por esta limitación es muy raro ver transmisores SINCRONOS en microcomputadores.

La transmisión ASINCRONA es la más común de los dos tipos. Esta no depende del clock hasta después de haber detectado el bit de START. Ese

bit de START se utiliza para comandar que se inicia la transmisión de un carácter y también se puede utilizar para determinar la longitud del tiempo de bit.

El número de bits en el carácter puede variar, el código original de 5 bits o BAUDOT es el código utilizado en teletipos y es el que más facilidades nos ofrece para la conexión con ellos, internacionalmente el telex utiliza 5 bits. El código BAUDOT de 6 bits es utilizado en algunos casos, pero el código de 7 bits (usualmente ASCII) es el más utilizado comúnmente.

La utilización de un código de 8 bits puede efectuarse de dos maneras completamente diferentes, la primera es utilizar un código de siete bits y el octavo utilizarlo como bit de paridad y la segunda es utilizar un código de 8 bits, por ejemplo el ASCII extendido.

Además en el formato de información enviado existe un bit de paridad que puede ser par, impar o no utilizarse y que no tiene nada que ver con la paridad generada para el octavo bit de dato como comentábamos antes.

El bit que se envía al final de estos caracteres ASINCRONOS es el bit de STOP. La longitud de este bit puede variar entre 1, 1.5 x 2 tiempos de bit. Este bit es siempre un nivel MARK ("1") y finaliza con el bit de START del siguiente carácter (Ver figura 2).

La mayor ventaja en la comunicación ASINCRONA es la relativa facilidad de uso y el hecho de no requerir grupos de caracteres. Esto es particularmente conveniente para el uso con microcomputadores donde es esencial el espacio de memoria. La principal desventaja de la comunicación ASINCRONA es que la paridad es la única protección contra cualquier error.

Señales de control

Existen 3 grupos de señales de con-



INTERFAS RS-232C
ASIGNACIÓN DE PINELES (PIN)

FIG. 5

tról. Para variaciones sobre estos grupos será necesario consultar los manuales técnicos de conexión de los aparatos.

GRUPO 1:

- PIN 1- PROTECTIVE GROUND (PG)
- PIN 2- TRANSMIT DATA (TD)
- PIN 3- RECEIVE DATA (RD)
- PIN 4- REQUEST TO SEND (RTS)
- PIN 5- CLEAR TO SEND (CTS)
- PIN 7- SIGNAL GROUND (SG)

Nota: CLEAR TO SEND Y REQUEST TO SEND se pueden conectar algunas veces entre sí, cuando se utilizan interfaces estándar.

GRUPO 2:

Consulte en el grupo 1 más:

- PIN 6- DATA SET READY (DSR)
- PIN 8- DATA CARRIER DETECT (DCD)
- PIN 20- DATA TERMINAL READY (DTR)

El grupo 2 es usado principalmente, en conexiones entre terminales y CPU's o en transmisiones entre CPU y CPU.

GRUPO 3:

Consiste en los otros dos grupos más:

- PIN 22- RING INDICATOR (RI)
- Este grupo se encuentra utilizado normalmente en estaciones fijas de MODEM's AUTO-ANSWER.

En la figura 3 tenemos un diagrama que representa un conector RS-232C con asignación de pines.

PIN 1- PROTECTIVE GROUND

Esta es la masa de chasis o masa de seguridad. Cuando se utiliza cable apantallado la malla se ha de conectar a este pin.

PIN 2- TRANSMIT DATA

Es por este pin por donde se envían los datos al exterior. Esta señal está a nivel MARK ("1") cuando no se utiliza. Antes de una transmisión, REQUEST TO SEND, CLEAR TO SEND, DATA SET READY y DATA TERMINAL READY han de estar a nivel MARK ("1").

PIN 3- RECEIVE DATA

Es por este pin por donde se recibe el TRANSMIT DATA enviado por otra unidad.

PIN 4- REQUEST TO SEND

Esta señal se pone a nivel alto cuando el interface desea transmitir datos.

PIN 5- CLEAR TO SEND

Indica que el data set o el MODEM está listo para transmitir. Esta señal es generada en respuesta de REQUEST TO SEND. En conexión con periféricos se conectará directamente a la señal REQUEST TO SEND.

PIN 6- DATA SET READY

Indica que el data set o el MODEM está listo para enviar y/o recibir datos. Esta señal se unirá con el DATA TERMINAL READY en interface con periféricos.

PIN 7- SIGNAL GROUND

A este pin se conectarán todas las masas lógicas.

PIN 8- DATA CARRIER DETECT

Indica la presencia en la línea de la señal portadora (usualmente se utiliza para la comunicación vía MODEM).

PIN 9- Reservado para DATA SET TESTING.

PIN 10- Reservado para DATA SET TESTING.

PIN 12- SECONDARY RECEIVE LINE DETECTOR

Esta línea es el canal secundario y se utiliza en altas velocidades para transmitir mensajes o status o en dirección opuesta al flujo principal de datos. El canal secundario no transmite nunca en la misma dirección que el canal primario, al mismo tiempo.

PIN 13- SECONDARY CLEAR TO SEND.

Es lo mismo que CLEAR TO SEND pero en canal secundario. Ninguno de los canales denominados secundarios puede ser utilizado en sistemas funcionando a más de 9600 baudios.

PIN 14- SECONDARY TRANSMITTED DATA

Igual que TRANSMIT DATA pero en canal secundario.

PIN 15- TRANSMISSION SIGNAL ELEMENT TIMING

Es el clock de transmisión utilizado en comunicaciones SINCRONAS.

PIN 16- SECONDARY RECEIVED DATA

Es lo mismo que RECEIVED DATA pero en canal secundario.

PIN 17- RECEIVED SIGNAL ELEMENT TIMING

Es el clock de transmisión recibido desde el exterior en comunicaciones SINCRONA.

PIN 19- SECONDARY REQUEST TO SEND

Es lo mismo que REQUEST TO SEND pero en canal secundario. Nótese que esta señal no aparece mientras la señal REQUEST TO SEND está a "ON".

PIN 20- DATA TERMINAL READY

Esta señal se utiliza para indicar que el data set de este "device" está listo para recibir y/o (depende del REQUEST TO SEND) transmitir datos.

PIN 21- SIGNAL QUALITY DETECT

Indica (cuando está OFF) que la portadora en la línea está en tal estado que probablemente habrá que repetir la transmisión.

PIN 22- RING INDICATOR

Se utiliza en MODEM's AUTO-ANSWER para indicar llamada por vía telefónica.

PIN 23- DATA SIGNAL RATE DETECTION

Cuando en la línea está instalado un MODEM con dos velocidades, esta señal se utiliza para seleccionar una u otra.

PIN 24- TRANSMIT SIGNAL ELEMENT TIMING

Por este pin se permite la entrada de un reloj externo para la velocidad de transmisión.

PIN 11- Señal sin asignar.

PIN 18- Señal sin asignar.

PIN 25- Señal sin asignar.

(Continuará)

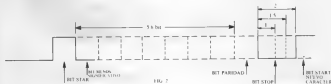


FIG. 7

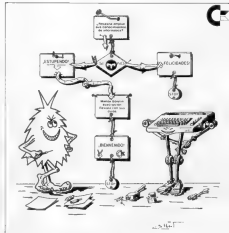
Clave para interpretar los listados de CLUB COMMODORE

Todos los listados que se publican en esta sección han sido ejecutados en el modelo correspondiente de la gama de ordenadores COMMODORE. Para facilitar la edición de los mismos en la sección y para mejorar su legibilidad por parte del usuario, se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a los ordenadores VIC-20 y COMMODORE 64, en los que se usan frecuentemente las posibilidades gráficas del teclado, se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes [] que indican la secuencia de teclas que se deben pulsar para obtener el carácter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes.

- [LRSR] Tecla cursor hacia abajo (con SHIF)
- [CRSR] Tecla cursor hacia arriba (con SHIF)
- [CRSR] Tecla cursor a la derecha (con SHIF)
- [CRSR] Tecla cursor a la izquierda (con SHIF)
- [HOME] Tecla CTRHOME (con SHIF)
- [CTR] Tecla CTRHOME (con SHIF)

Las indicaciones [BLK] a [YU] corresponden a la pulsación de las teclas de la 8ª fila a la tecla CTR. Lo mismo sucede con [RVSON] y [RVSON] respecto a la tecla CTR y las teclas 9 y 10.

El texto de las indicaciones constan de la parte COMM o SHIF seguida de una letra, número o símbolo —por ejemplo [COMM+] o [SHIF A]—, lo que indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIF y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior COMMODORE y la tecla A.



COMMODORE y la SHIF y A respectivamente.

En los signos gráficos además se cuenta el número de veces que apor-

tece. Por ejemplo, CTRSR, que vale a 7 pulsaciones de la tecla cursor a la derecha y YU (tres pulsaciones de la barra espaciadora).

EJEMPLARES ATRASADOS DE «CLUB COMMODORE»

Para poder satisfacer la creciente demanda de números atrasados de nuestra Revista, agotada en todas sus ediciones, hemos puesto en marcha un Servicio para suministrar fotocopias de los ejemplares que nos sean solicitados. Para recibir las fotocopias de una o de varias ediciones, no hay más que enviarnos el boletín con los datos indicados.

SERVICIO DE FOTOCOPIAS

NUMERO DE LA EDICION SOLICITADA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(Poner una X debajo del número de edición pedido)

Peticionario: D

Calle

Población

Provincia

Forma de envío: contra reembolso

La colección completa del 0 al 15: 2.500 Ptas.

Precio de la edición fotocopiada: 250 ptas.

SOFTWARE COMMODORE 64

se puede utilizar su C 64
solamente para pagar
quedará no necesita estos
programas.

PROGRAMAS EN DISCO

BASE DE DATOS

1000 ARTÍCULOS EN SU BASE DE DATOS

- Sistema de trabajo con 10 operaciones
- Consultas por pantalla e impresoras
- Búsqueda por campos
- Listados de ficheros
- Listados por el contenido del fichero
- Etc.

GESTIÓN STOCK

1000 ARTÍCULOS

1000 SPELTES POR PERÍODO

- Consultas por pantalla e impresoras
- Impresoras de precios de compra y precios de venta
- Tarjetas de gestión
- Listados por ficheros de gestión
- Listado del fichero principal completo
- con todos los depósitos del periodo
- Listado por periodo
- Etc.

GESTIÓN COMERCIAL

999 ARTÍCULOS

999 CLIENTES

99 REPRESENTANTES

99 ZONAS

99 BANCOS

999

- Cuentas de abonos y facturas
- Sistema de ventas
- Confirmación de pedidos negociados
- Remesas bancas
- Control albaranes
- Registro de notas
- Gestión de contratos
- Liquidaciones
- Listados de todos los ficheros
- Etc.

9 May 1985

CONTABILIDAD

pida información y... **PONGA EL "64" A TRABAJAR EN SERIO**

con nuestros programas **PUEDA !**

EAF

microgestion

compra de correo 563 563, 41 5
Barcelona 13
tel 93 231 66 87
apartado 24 143



Una Excursión en Basic Más allá del Manual

Por Jeffrey Mills

Aquí presentamos un salvavidas para el entusiasta que se acaba de comprar un Commodore 64. Este es el primero en una serie de artículos que le ayudará a navegar por los rápidos y los remolinos de la programación del Basic.

Así que te has leído el manual. Además, has programado los ejemplos del manual. Pero cómo vas a encajar toda esta información? Vamos a ver algunas de las cosas que tú y tu Commodore 64 podéis hacer.

El primer paso es aprender a utilizar el ordenador de manera eficaz: el de saber programarlo para que satisfaga tus propias necesidades. ¿Y qué es un programa? Una definición simplificada podría ser: Un programa es un conjunto de instrucciones detalladas que le indican al ordenador, paso a paso, cómo realizar una serie de tareas, en un orden lógico, para poder alcanzar una meta.

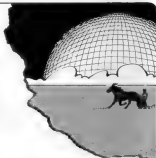
“Pero como empiezo?” estarás diciendo. Pues, primero vamos a hablar de las normas básicas para la preparación previa y luego empezaremos a desarrollar un programa.

Comandos para Operaciones Preparatorias

1) **New (Nuevo)**—El comando “NEW” es muy potente, y se tiene que usar con cuidado. Cuando quieres iniciar un programa nuevo, “New” borra el programa anterior almacenado en la memoria, además de las variables para dicho programa. Mas adelante consideraremos las variables.

Pero ojo. Una vez introducido el comando “NEW”, no es posible recuperar el programa anterior, así que hay que salvarlo antes de utilizar este comando si no lo quieres perder. Para refrescar la memoria sobre New, consulta la página 115 en la “Guía para Usuarios del Commodore 64”.

2) **CLR**—Este comando “despeja” la memoria del ordenador sin modificar ni borrar cualquier programa que se haya almacenado allí. Todas las variables se vuelven a poner a cero, y los punteros se vuelven a sus estados iniciales. (Los punteros son las direcciones de las diversas cosas importantes que el ordenador tiene que recordar mientras se está ejecutando un programa). El tema de CLR se trata en la página 117 de la Guía del Usuario.



3) **LIST (Listado)**—Este comando te permite ver un listado del programa que estás procesando. Para parar el comando “List”, utiliza la tecla run/stop.

Puedes ejecutar “List” para que sólo aparezca una parte determinada. Esto se hace especificando los números de las líneas que quieres ver. Por ejemplo,

- **LIST 100-200** (se presentan todas las líneas de 100 hasta 200), o

- **LIST -999** (se presentan todas las líneas hasta inclusive la línea 999), o

- **LIST 345** (sólo se presenta la línea 345).

Si se da el comando “List” sin especificar ningún número de línea, se presenta en listado el programa entero. El comando “List” se define con ejemplos en la página 115 en la Guía del Usuario.

4) **Line Numbering (Numeración de Líneas)**—Cuando se escribe un programa, el ordenador necesita saber el orden en que tiene que realizar las diversas tareas. Esto se realiza mediante los números de líneas, que pueden abarcar desde 0 hasta 65535.

Al numerar un programa, es mejor contar en decenas (10, 20, 30, 40...). Con este método se puede volver atrás para insertar una línea olvidada. Por ejemplo, podría resultar necesario insertar algo entre la primera línea (10) y la segunda (20). La nueva línea podría en este caso numerarse 15. Si las líneas se hubieran numerado 1, 2, 3, 4, otra línea no se podía haber introducido entre 1 y 2.

5) **REM**—Este comando, que introduce un comentario explicativo, no afecta en absoluto la operación del programa, sino que aclara simplemente lo que el programa tiene que hacer en este punto. Esta documentación es bastante importante, ya que con el tiempo se olvida fácilmente lo que hay que hacer.

Lo primero que se aprende a utilizar en un programa real es la sentencia **REM**. Los comandos que hemos explicado hasta ahora normalmente se introducen directamente a través del teclado. Las sentencias **REM** también constituyen la herramienta más útil para que el programador descubra la causa de mal funcionamiento de un programa. Por lo tanto, se deben usar con frecuencia para escribir un programa. A continuación, veremos un ejemplo.

```
10 REM *** THIS IS A REMARK  
(ESTO ES UN COMENTARIO).
```

Foma nota del uso de los asteriscos, que se colocan en la sentencia para que ésta resulte más visible cuando un programa se presenta en forma de listado. Aunque no sean imprescindibles, son útiles. **REM** se explica en la página 124 de la Guía del Usuario.

Los Primeros Pasos

Lo primero que hay que hacer para desarrollar un programa es decidir exactamente que es lo que te quieres que realice el ordenador. El ejemplo que te voy a presentar indica cómo el Commodore 64 guarda el listado de todos los programas, incluyendo los números de las cintas donde se almacenan.

Empezamos con un comentario que identifica el programa y su autor. Se teclea:

```
10 REM *** PROGRAMA/CATA-  
LOGO DE CINTAS ***  
20 REM *** ESCRITO POR tu  
nombre **
```

El segundo paso es el de despejar la pantalla, donde se va a representar el catálogo de programas. Esto se puede realizar desde dentro del programa utilizando un comando incluido en una sentencia **PRINT**.

La sentencia **Print** le indica al ordenador que tiene que representar algo directamente en pantalla, empezando a partir de la siguiente línea disponible. Si la pantalla está llena, el Commodore desplaza el texto hacia arriba para que la parte de abajo quede libre para incluir la línea que tiene que imprimirse.

Por lo tanto, la primera línea desaparece. Debido a esto, debes, de momento, limitar la lista a 15 líneas, para evitar este desplazamiento. Tratare este problema en otro artículo.

Para que se imprima un número en pantalla, se utiliza una línea como la siguiente:

```
30 PRINT 12
```

Cuando el ordenador llega a la línea 30 imprime el número 12 en pantalla. Para que se imprima una palabra se tiene que escribir entre comillas.

Por ejemplo:

```
30 PRINT "DOCE"
```

El ordenador imprimirá la palabra **DOCE** en la pantalla al llegar a la línea 30.

Al igual que se imprime una palabra, se puede imprimir cualquiera de las teclas de comando del teclado. Estos se llaman caracteres de comando incluido. Cuando se imprimen, realizan las tareas que les han sido asignadas, como mover el cursor (que indica la posición donde se imprimirá la siguiente unidad de datos) o borrar la pantalla.

Para borrar la pantalla se teclea:

```
30 PRINT "1[Shift]-CLR/HOME1"
```

Si se escribe esa tecla entre comillas la pantalla no se borra en seguida. Lo que aparece es un carácter gráfico de "corazón" invertido. Su presencia borra la pantalla en el momento apropiado mientras se ejecuta el programa.

(Cuando se intercala un comando, siempre se representa como un carácter invertido distinto para cada tecla).

En el manual no se tratan los comandos incluidos de teclado. Tardarás un poco en aprender los caracteres invertidos que representan las teclas de comando. Sin embargo, eso se aprende mediante la práctica.

Resulta útil saber que para introducir un programa (o una sentencia directa) mediante el teclado, se puede usar una interrogación para representar la palabra **PRINT**. Cuando se presenta el listado del programa, saldrá la palabra **PRINT**. (Existen varias palabras abreviadas que se presentan en una tabla en la página 130-131).

La sentencia **PRINT** se explica en el manual en las páginas 22-29 y 123-124.

Encabezamiento y Espaciado

Cuando el ordenador termina de borrar la pantalla, se puede imprimir un encabezamiento.

Existen varios métodos para centrar el encabezamiento. De momento, es mejor incluir espacios en la sentencia **PRINT**. Se teclea:

```
40 PRINT " CATALOGO DE CIN-  
TAS"
```

Hay 14 espacios en una sentencia **PRINT**.

(¿Por qué 14? El encabezamiento ocupa 12 espacios y cada línea ocupa 40 espacios en la pantalla. Quedan 28

espacios que se dividen en partes iguales entre los dos lados).

Para que quede una línea en blanco después del encabezamiento y antes de la lista del catálogo, se imprime una línea vacía.

```
50 PRINT
```

Antes de hablar del listado, es útil conocer algunos de los métodos más sencillos para que la representación en pantalla quede ordenada.

Existen dos formas sencillas para dejar espacios en la salida en pantalla: la coma y el punto y coma. Si se utiliza el punto y coma, la salida no tendrá ningún espacio entre un campo y el siguiente. (El campo es un término que se emplea para indicar un grupo de caracteres que forman parte de un registro entero. Por ejemplo, el número de cinta en el programa actual constituye un campo; el nombre del programa constituye otro. Los dos campos combinados forman el registro completo).

Si se imprime la salida con comas, quedará espaciada en la pantalla en lo que se llaman zonas de impresión. Las zonas de impresión del C-4 tienen una longitud de diez espacios. Cada campo empieza en la siguiente zona de impresión que queda disponible. A continuación se presentan un par de ejemplos.

Supongamos que hay dos programas que se tienen que incluir en el listado del catálogo. Se llaman el Juego 1 y el Juego 2, y se salvan en la cinta 101. Los dos programas salen en el catálogo mediante las sentencias de impresión. Se teclea:

```
60 PRINT "101"; "GAME 1" (Juego 1)  
70 PRINT "101"; "GAME 2" (Juego 2)
```

La línea 60 se imprime con un punto y coma mientras que la línea 70 se imprime con una coma.

Si se ejecuta el programa se observa la diferencia entre las dos líneas.

La primera no contiene ningún espacio entre los campos, mientras que la segunda se imprime en zonas individuales.

Si la entrada de "número" en la línea 70 hubiera contenido más de diez caracteres, el nombre del programa se habría impreso al principio de la siguiente zona de impresión disponible (en este caso, la columna 20 en vez de la columna 10).

Ahora se teclea la línea 60 para que se modifique y se parezca a la línea 70: **60 PRINT "101"; "GAME 1" (Juego 1)**

Ambas líneas quedarán espaciadas de modo uniforme al ejecutarse el programa.

Ahora que sabes teclear una línea de programa con una sentencia **PRINT** diferente para cada entrada en el catálogo, puedes seguir y terminar tu lista.

Ficheros en disco (5)

Técnicas de Acceso Directo

3. Distribución de los ficheros en disco:

Siguiendo el índice marcado en el artículo III de la serie, a continuación vamos a ver cómo se distribuyen los ficheros en el diskette, profundizando sobre lo visto en el artículo III, apartado 1.

En todos los discos existen tres tipos diferentes de bloques de control, que son:

- A) CABCERA DEL DISKETTE (Directory Header)
- B) BLOQUES DEL BAM
- C) BLOQUES DEL DIRECTORIO

Las funciones del BAM y el directorio ya fueron descritas en el artículo anteriormente mencionado. Veamos pues para qué nos sirve la cabecera del diskette:

Manuel AMADO

A) CABCERA DEL DISKETTE

Cuando se inicializa un diskette, o bien se empieza a trabajar en él sin inicializarlo (caso de 4040, 8250 y 8050) que se inicializan automáticamente el DOS se sitúa siempre, en un determinado sector del diskette, dependiendo este sector exclusivamente del modelo

de floppy empleado. Esta cabecera contiene una serie de parámetros que definen al diskette y en su caso los parámetros necesarios con el primer bloque de BAM y del directorio para poder acceder y procesar en su caso la información contenida en el diskette.

En la fig. 1 pueden observarse las cabeceras de diskette correspondiente a los floppys IBM: modelos 4040, 1540, 1541 y 8050.

Observamos la diferente estructura de las cabeceras de disco entre el 4040 (es igual en los 1540 y 1541) y el 8050/8250. En primer lugar, al sólo

tener que controlar el 4040 663 bloques, es suficiente disponer de 139 bytes de BAM para controlar su disponibilidad. Por ello, en el 4040 se aprovecha el sector correspondiente a la pista 18 sector 0 para ubicar en él el BAM y la cabecera del disco. Por el contrario, en el caso del 8050 se necesitan continuos 2083 bloques con lo que son necesarios dos bloques de BAM para controlarlos y consecuentemente, la cabecera del disco y los bloques de BAM están separados y en la cabecera del disco se hace necesario un puntero al primer sector del BAM.

FIGURA 1
DIRECTORY HEADER- 4040, 1540 y 1541

Pista 18, Sector 0		
BYTE	CONTENIDO	DEFINICION
144-161		Nombre del disco relleno con espacios shifted
162-163		ID del Disco
164	160	Espacios -shifted-
165, 166	50,65	Representación ASCII para 2A que es la versión y tipo de formato del DOS
166-167	160	Espacios shifted
171-255	0	Nulos, no se usan

NOTA: Los caracteres ASCII pueden aparecer de la posición 180 a la 191 en algunos diskettes.

BLOQUE - DIRECTORY HEADER- DEL 8050

Pista 39, Sector 0		
BYTE	CONTENIDO	DEFINICION
0,1	38,0	Pista y sector del primer bloque del BAM
2	67	El carácter ASCII C al formato 8050
3	0	Señal nula para posteriores usos del DOS
4,5	0	No se usan
6-21		Nombre del disco relleno con espacios shifted
22,23	160	Espacios shifted
24,25		ID del disco
26	160	Espacios shifted
27,28	50,67	Representación en ASCII del 2C que es versión y tipo de formato del DOS
29-32	160	Espacios shifted
33-255	0	No se usan



La estructura en el modelo 8250 es análoga a la del 8050.

El resto de información contenida en las cabeceras se deduce de la simple observación de la figura 1.

B) ESTRUCTURA DE LOS BLOQUES DEL BAM;

En el caso de la estructura de los bloques de BAM del 4040 y del 8050/8250, las diferencias apreciadas

son notables con respecto al caso anterior (ver fig. 2). Una característica importante y común a ambos modelos es que los dos primeros bytes del último bloque de BAM contienen el puntero o link al primer bloque del directorio.

En el caso del 4040, la estructura en sí del BAM (lo que es el mapa de bits propiamente dicho), no tiene ninguna complicación, ya que cabe enteramente en la primera parte de la cabecera del disco y está distribuido por pesos, o sea, el bit 343 nos informa de la disponibilidad del sector o bloque número 343 del disco.

Para el 8050, se complica esta estructura ligeramente, ya que tiene que distribuir el BAM entre dos bloques del disco, 36-0 y 36-1. El primero controla los sectores comprendidos entre la pista 1 y 50, conteniendo en los dos primeros bytes un puntero a la pista y sector del siguiente bloque de BAM. El segundo bloque del BAM contiene en los dos primeros bytes el puntero o link al primer bloque del directorio, y controla el resto de los pistas del 8050 (51-77). Además, se repite en cada bloque el carácter 63 indicador de que se trata del formato 8050. La estructura de los bloques de BAM del 8250 es análoga a la del 8050, con la diferencia de que al tener que controlar 4133 bloques (prácticamente el doble del 8050), se necesitan 4 bloques de BAM.

C) BLOQUES DEL DIRECTORIO

El directorio del disco está formado por los nombres y direcciones de los distintos ficheros que componen este disco. Su estructura es común a todos los floppys CBM, y cada bloque contiene en los dos primeros bytes la dirección del siguiente bloque del directorio, comenzando 0 y 255 (0 y 511 en hexadecimal) en el caso de que sea el último bloque. El resto del bloque corresponde a las ocho entradas de fichero que puede llegar a contener dicho bloque.

En la figura 3 observamos que para cada fichero existe su entrada correspondiente en el directorio. Pues bien, el contenido de estos bytes que en la figura 3 se representa como "entrada fichero X", depende del tipo de fichero que describe.

Resumiendo, los bloques del directorio forman una estructura encadenada o lista, en la que cada uno de sus elementos apunta o está ligado al siguiente mediante un puntero o link. Estos bloques forman como una especie de índice de los ficheros del disco, pues nos indican en qué lugar del disco se halla cada fichero.

La estructura de la entrada de fichero del directorio se puede observar en la figura 4.

FIGURA 2
FORMATO DEL BAM DEL 8050

Primer bloque del BAM: Pista 38, Sector 0		
BYTE	CONTENIDO	DEFINICION
0,1	38,3	Pista y sector del segundo bloque del BAM
2	67	Carácter ASCII C al formato del 8050
3	0	Señal nula por futuros usos del DOS
4	1	El menor número de pista representado en este bloque del BAM
5	51	El mayor número de pista + 1 en este bloque del BAM
6		Número de bloques no usados en la pista 1
7-10		Mapa de representación de bit de bloques disponibles en la pista 1
11-255		* BAM para las pistas del 2 al 50 5 bytes por pista
Segundo bloque del BAM: Pista 38, Sector 3		
BYTE	CONTENIDO	DEFINICION
0,1	38,1	Pista y sector del primer bloque del directorio
2	67	Carácter ASCII C formato de 8050
3	0	Señal nula para usos posteriores del DOS
4	1	El menor número de pista representado en este bloque BAM
5	51	El mayor número de pista + 1 en este bloque BAM
6		Número de bloques no usados en la pista 51
7-10		Mapa de representación de bit de los bloques utilizables en la pista 51
11-140		* BAM para pistas 52-77, 5 bytes por pista
141-255		No usado

* ESTRUCTURA DE ENTRADA DE BAM PARA UNA PISTA

BYTE	DEFINICION
0	número de sectores disponibles para pista
1	mapa de bits de los sectores 0-7
2	mapa de bits de los sectores 8-15
3	mapa de bits de los sectores 16-23
4	mapa de bits de los sectores 24-31

1 disponible
 0 no disponible

NOTA: -BLOCKS FREE- puede aparecer en las posiciones 180 a la 191 en algunos disquetes.

FIGURA 2
FORMATO DEL BAM DEL 4040

Pista 18, Sector 0		
BYTE	CONTENIDO	DEFINICION
0,1	18,01	Pista y sector del primer bloque del directorio
2	65	El carácter ASCII A indicando el formato del 4040

3	0	Señal nula para posteriores utilidades del DOS
4-143		* Mapa de bit de bloques disponibles para las pistas 1-35 (BIT MAP)
* 1 = bloque disponible 0 = bloque no disponible cada bit representa un bloque.		

FIGURA 3
FORMATO DEL DIRECTORIO

Pista 18, Sector 1 para el 4040, 1540 y 1541 Pista 39, Sector 1 para el 8030		
BYTE	DEFINICIÓN	
0,1	Pista y sector del siguiente bloque directorio	
2-31	* Entrada de fichero 1	
34-63	* Entrada de fichero 2	
66-95	* Entrada de fichero 3	
98-127	* Entrada de fichero 4	
130-159	* Entrada de fichero 5	
162-191	* Entrada de fichero 6	
194-223	* Entrada de fichero 7	
226-255	* Entrada de fichero 8	

* Ver Figura 4

FIGURA 4
ESTRUCTURA DE LA ENTRADA DEL DIRECTORIO SIMPLE

BYTE	CONTENIDO	DEFINICIÓN
0	128 = tipo	Tipo de fichero OR'ed con #80 para indicar que el fichero ha sido correctamente cerrado TIPOS 0 Borrado (DEL) 1 Secuencial (SEQ) 2 Programa (PRG) 3 Usuario (USR) 4 Relativo (REL)
1,2		Pista y sector del primer bloque de datos
3-18		Nombre del fichero relleno con espacios Shifted
19,20		Sólo ficheros relativos: pista y sector para el primer bloque de sector
21		Sólo ficheros relativos: longitud de la ficha
22-25		No usados
26-27		Pista y sector del fichero de reemplazo cuando un OPEN @ entra en efecto
28-29		Número de bloques en el fichero menor byte, mayor byte (peso bajo, peso alto)

FIGURA 5
FORMATO DE UN FICHERO DE PROGRAMA

BYTE	DEFINICIÓN
0,1	Pista y sector del siguiente bloque en el fichero de programa
2-255	254 bytes de información del programa almacenado en el formato de memoria en el VIC. (Con las palabras clave bajo forma simbólica). El final de fichero está marcado con tres bytes iguales a 0

En ella hay el tipo de fichero que es, el puntero al primer bloque de datos del fichero, el nombre del fichero, el número de bloques que ocupa el fichero y una serie de parámetros propios de los ficheros relativos. La estructura de los bloques de datos para los ficheros secuenciales y programa puede verse en la figura 5. En el caso de ser un fichero programa, los dos primeros bytes de información del primer bloque del fichero contienen la dirección de carga en memoria de dicho programa.

Consecuentemente, en los ficheros secuenciales y programa, su distribución es semejante en ambos casos: cada bloque de datos contiene en los dos primeros bytes el puntero al siguiente bloque (0, 255 si es el último), y los 254 bytes restantes contienen los bytes de datos o información propios del fichero. Observamos pues que ambos tienen una estructura parecida a la de los bloques del directorio, o sea, una lista o cadena de bloques.

Y ahora me haréis una buena pregunta: pues muy bien, muy bonito, pero, ¿para qué sirve toda esta documentación? Pues es bien sencillo y complicado a la vez de contestar. Para varias cosas, y sobre todo dependiendo de la imaginación que le pongáis al asunto. Vamos a ver solamente dos aplicaciones sencillas: los agujeros la idea, más adelante se plasmarán en programas, o si os animáis y escribís a la redacción los programas basados en alguna de estas dos ideas, se publicarán las mejores versiones de cada idea).

A) Supongamos que se ha estropeado un determinado bloque del disco, pues nos da al acceder a él un error 23, READ ERROR, por ejemplo, al leer los datos en un fichero secuencial. Pues bien, podemos intentar recuperar parte de los datos del fichero siguiendo la cadena de ficheros encañados por acceso directo, y marcando el anterior bloque al que da error como fin del fichero, o bien, si localizamos el bloque siguiente (suponiendo que no esté estropeado) al bloque que nos da error, podemos reconstruir la cadena de bloques linkados haciendo un puente entre el anterior bloque al error y el posterior, con lo cual sólo habremos perdido un bloque de información del fichero.

B) Hacer un utilitario de copia selectiva de ficheros de un diskette a otro del mismo floppy, mediante exploración del contenido del directorio del diskette a copiar, y seleccionando el operador sólo aquellos ficheros que se deseen copiar.

Para finalizar deseo hacer constar que la información sobre los ficheros relativos fue publicada en los números 9, 10 y 11 de nuestra revista. ■

Video Casino

Cada mes, COMMODORE WORLD, presentará programas de juegos originales que vd. puede usar y disfrutar con su micro. El primero de la serie se titula "Tiro al Blanco".

La galerna de tiro al blanco ha sido, y sigue siendo, un juego muy popular. Requiere la coordinación entre la mano y el ojo en contra de los caprichos del blanco que se mueve al azar.

Algunos de los primeros juegos para el ordenador eran adaptaciones de los muchos juegos de tiro al blanco. A veces el participante era un vaquero en video. Hoy en día, los blancos pueden ser naves que navegan sobre la superficie de un océano en la pantalla, por ejemplo, o masas peligrosas de cera gaseosa.

"Tiro al Blanco" presenta un blanco que se mueve rápidamente. Una capa se desplaza por la parte superior de la pantalla de izquierda a derecha. Al pasar, va descendiendo cada vez más, hasta que al final el estanco chocó con su nave localizada en la parte inferior de la pantalla. Vd. puede usar el mando (joystick) para mover la nave de un lado para otro, para seguir el paso del blanco así se desea. En el momento más oportuno, al pulsar el botón de disparo, se lanza un misil.

En este momento, la nave no se mueve, y el mando (joystick) controla el misil. Vd. acerca el proyectil al blanco enemigo. Si acierta en el blanco, se desencadena una explosión impresionante. Si lo hace con cuidado puede acertar dos, tres o más veces en una sola pasada, acumulando una puntuación realmente alta.

Variable Útil

El programa está dividido en varios módulos. El primer paso después de las instrucciones es el de inicializar ciertas variables que se utilizarán más adelante en el programa. Al definir números que se utilizarán una y otra vez como variables, la operación del

programa se hará mucho más rápida. En vez de evaluar la constante 6, por ejemplo, cada vez que el programa la encuentra, el ordenador utiliza la variable B1000000.

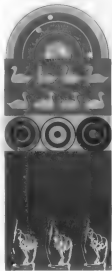
El ordenador comprueba esta variable en una tabla interna rápidamente, dado que todas las variables se almacenan en el orden en que el programa las utiliza. Dado que B1000000 fue definido en la línea 200, se encuentra al principio de la tabla de verificación de variables. En vez de hacer un POKI en una posición en la memoria de color con el 6 para que se vuelva azul, se puede hacer un POKI de la variable B1000000, que es más fácil de recordar que el número seis.

De forma parecida, en vez de hacer un POKI 32 en la memoria de pantalla para conseguir un espacio (CTRY 132), simplemente se hace un POKI de la variable SPVCL, definida en la línea 230. También se definen dos variables adicionales, CSRRLN y CTAR, pero debido a otros motivos.

Como vd. bien sabe, el VIC-20 almacena un registro de lo que aparece en la pantalla en una serie de 508 posiciones de memoria. El color del carácter de cada una de dichas posiciones también se almacena en 508 posiciones de memoria. Si se cambia el contenido de cualquiera de estos bytes se cambia el carácter en dicha posición de pantalla, o se cambia su color.

En el VIC-20 no ampliado, los códigos para los caracteres de pantalla se almacenan a partir de la posición de memoria 3600 y el mapa de memoria del código de color comienza a partir del 8400. Desafortunadamente, estas posiciones de memoria pueden cambiar, según la cantidad de memoria añadida al VIC-20. Por eso

muchos de los juegos en venta especifican que se usan solamente con un VIC-20 no ampliado.



```

1 REM VIC-20 VERSION
10 REM *****
20 REM *
30 REM *TIRO AL BLANCO*
40 REM *
50 REM *****
60 PRINT"[SHIFT CLR][CRSR DN][CRSR DN]"
70 PRINTTAB(4)"[CTRL 9][CTRL 3]TARGET SHOOT[CRSR DN][C
  RSR DN]"
80 PRINTTAB(1)"USE JOYSTICK TO MOVE"
90 PRINTTAB(1)"YOUR BASE ON BOTTOM"
100 PRINTTAB(1)"OF SCREEN, AND TO "
110 PRINTTAB(1)"STEER YOUR ARROW.TRY"
120 PRINTTAB(1)"TO GET AS MANY HITS"
130 PRINTTAB(1)"AS POSSIBLE BEFORE"
140 PRINTTAB(1)"TARGET REACHES BOTTOM[CRSR DN][CRSR DN]"
150 PRINTTAB(6)"[CTRL 9][CTRL 3]HIT ANY KEY"
160 GET AS:IF AS="R" GOTO 160
170 POKE36879,104
180 PRINT"[SHIFT CLR]"
190 DF=30720
200 WHITE=1:CFAN=3:BLUE=6
210 RSPACE=160
220 ARROW=10
230 SPACE=32
240 CSCREEN=37888+4*(PEEK(36866)AND128):81=CSCREEN
250 CHAR=4*(PEEK(36866)AND128)+64*(PEEK(36869)AND128):B
  =CHAR:1=CHAR+484
260 DF=CSCREEN-CHAR
270 DD=37154
280 PA=37137
290 PR=37152
300 POKE 37139,0
310 GOTO 190
320 POKEDD,127
330 SJ=-(PEEK(PR)AND128)+0
340 POKE DD,255
350 P=PEEK(PA)
360 S2=-(PAAND16)+0
370 PR=-(PAAND12)+0
380 RETURN
390 PRINT"[SHIFT CLR]"
400 SH=CSCREEN
410 PO=E
420 HITS=0:BALLS=0
430 BALLS=0
440 GOSUB320
450 IF SH-DF-PO GOTO 870
460 IF S1=0 AND S2=0 THEN GOTO 530
470 IF F2=1 THEN S1=S1+S2+5:3:GOTO530
480 IF S1=1THENPO=PO+1:IF PO=2+22THENPO=1+22
490 IF S2=1THENPO=PO-1:IF PO=1THENPO=2
500 IF S2=-1 GOTO 520
510 POKE PO,63:POKEPO-1,32:GOTO530
520 POKE PO,65:POKEPO-1,32
530 SH=SH+1
540 IF F2=0 AND PR=1THENFL=1:LL=50:GOSUB1020
550 IF FLAG=1THENGOSUB620
560 IF F7=1THENGOSUB670
570 POKE SH,CFAN
580 POKESH DF,RSPACE
590 POKE SH-1,BLUE
600 POKESH-DF 1,32
610 GOTO 440
620 N1=PO-22:F2=1
630 FL=0
640 POKE N1,ARROW
650 N1=N1-22
660 RETURN
670 N=PEEK(N1)
680 IF N=BLTHEN GOSUB 1020:GOTO 700

```

Un programa que contiene una línea como la siguiente:

30 POKE 36881:POKE38400,6
produce una pelota azul en la pantalla de un VIC-20 no ampliado, pero no imprimirá nada en un VIC que tiene una ampliación de memoria mayor que el 3K.

Sin embargo, en vez de hacer un POKE directamente en una posición de memoria, se puede hacer un POKE en una posición relativa de memoria. Digamos que la memoria de caracteres comienza en la posición X, y la memoria de colores en la posición Y. Se puede hacer un POKE en una cuarta posición para cada uno tecleando POKE X+4 o POKE Y+4. Si definimos X e Y como los diferentes puntos de partida, según la cantidad de memoria, el mismo programa funcionará en cualquier VIC-20.

Esto es lo que se hace en "Tiro al Blanco". En vez de X e Y, usamos CSCREEN (para la memoria de colores) y CHAR (para la memoria de caracteres). Las líneas 240 y 250 hacen un PEEK en una posición que les indica la memoria disponible que les queda, y luego calculan las verdaderas posiciones para los respectivos mapas de memoria.

Para simplificar un poco más, se calcula la diferencia (DF) entre los dos puntos de partida. Así, se queremos hacer un POKE a la cuarta posición de cada mapa de memoria, simplemente tecleamos POKE CSCREEN + 4 y POKE CSCREEN + 4 + DF.

El motivo de hacer esto, por ejemplo, es que la posición de la nave (definida como SH) cambia a medida que el jugador manipula el mando (joystick). Tanto el carácter como su color se cambian en la nueva posición al hacer un POKE en los nuevos valores de SH y SH-DF con el carácter y el color.

EQUIVALENTES ESPAÑOL

```

1 REM VERSION EN ES
10 REM TIRO AL BLANCO
20 PRINTTAB(1)"ELIJA MANDO
  JOYSTICK PARA MOVER"
30 PRINTTAB(1)"U BASTA EN LA
  PARTE INFERIOR"
40 PRINTTAB(1)"DE LA PANTALLA
  Y PARA"
50 PRINTTAB(1)"GUIAR EL ELI
  CIRA INTENIA"
60 PRINTTAB(1)"ALINTARECOMAS"
70 PRINTTAB(1)"POSSIBLE ANTES DE
  QUE"
80 PRINTTAB(1)"BLANCO ELLEGE
  ARAIO(CRSR DN)(CRSR DN)"
90 HIT ANY KEY "PUSE CUAL-
  QUIER TECLA"
100 NEW HIGH SCORE = "NUEVA
  MAXIMA PUNTUACION"
110 FLAG AGAIN = "JER GAS OTRA
  VEZ"

```




```

90 PRINTAB(8)"FOUR BASE ON BOTTOM"
100 PRINTAB(8)"OF SCREEN, AND TO "
110 PRINTAB(8)"STEER YOUR ARROW, THE "
120 PRINTAB(8)"TO GET AS MANY HITS "
130 PRINTAB(8)"AS POSSIBLE BEFORE"
140 PRINTAB(8)"TARGET REACHES BOTTOM(CRSE BN) (CRSE EN)
150 PRINTAB(8)"(CRSE BN) (CRSE BN) PLUS JOYSTICK INTO PO
160 PRINTAB(8)"CLOSEST TO POWER SWITCH. (CRSE BN) (CRSE

```

EQUIVALENTES ESPAÑOL

18	-50 REM TIRO AL BLANCO-64	200	PUNTAIRI "GUITAR TU MISIL
190	PUNTAIRI "UTILIZAR MANDO	210	PUNTAIRI "ACERTAR LOMAS"
195	PUNTAIRI "PARA MOVER"	220	PUNTAIRI "POSSIBLE ANTES DE
196	PUNTAIRI "TU BASE EN LA	230	PUNTAIRI "BLANCO LLUEGE
197	PUNTAIRI "PARTE INFERIOR"	240	PUNTAIRI "CON CERA DMI"
200	PUNTAIRI "DE LA PANTALLA,	250	PUNTAIRI "JUNTAIRI =
201	Y PANA"	260	PUNTAIRI "MANDO LUYOISITE"
202	PUNTAIRI "GUITAR TU MISIL	270	PUNTAIRI "EN SU CONECTOR"
210	PUNTAIRI "ACERTAR LOMAS"	280	PUNTAIRI "POWER SWITCH"
220	PUNTAIRI "POSSIBLE ANTES DE	290	PUNTAIRI "MAS CERCA A FUENTE DE
230	PUNTAIRI "BLANCO LLUEGE		ENERGIA"
240	PUNTAIRI "CON CERA DMI"		
250	PUNTAIRI "JUNTAIRI =		
260	PUNTAIRI "MANDO LUYOISITE"		
270	PUNTAIRI "EN SU CONECTOR"		
280	PUNTAIRI "POWER SWITCH"		
290	PUNTAIRI "MAS CERCA A FUENTE DE		
	ENERGIA"		

```

1800 N1=40
1820 RETURN
1900 H=PEK(N1)
2000 IFB=BITEN COSUB 1030 GOTO 730
2100 IF W=SPACE THEN GOSUB 800
2200 FOR N1=1 TO N
2300 FOR N1=1 TO N
2400 FOR N1=1 TO N
2500 FOR N1=1 TO N
2600 FOR N1=1 TO N
2700 FOR N1=1 TO N
2800 FOR N1=1 TO N
2900 FOR N1=1 TO N
3000 FOR N1=1 TO N
3100 FOR N1=1 TO N
3200 FOR N1=1 TO N
3300 FOR N1=1 TO N
3400 FOR N1=1 TO N
3500 FOR N1=1 TO N
3600 FOR N1=1 TO N
3700 FOR N1=1 TO N
3800 FOR N1=1 TO N
3900 FOR N1=1 TO N
4000 FOR N1=1 TO N
4100 FOR N1=1 TO N
4200 FOR N1=1 TO N
4300 FOR N1=1 TO N
4400 FOR N1=1 TO N
4500 FOR N1=1 TO N
4600 FOR N1=1 TO N
4700 FOR N1=1 TO N
4800 FOR N1=1 TO N
4900 FOR N1=1 TO N
5000 FOR N1=1 TO N
5100 FOR N1=1 TO N
5200 FOR N1=1 TO N
5300 FOR N1=1 TO N
5400 FOR N1=1 TO N
5500 FOR N1=1 TO N
5600 FOR N1=1 TO N
5700 FOR N1=1 TO N
5800 FOR N1=1 TO N
5900 FOR N1=1 TO N
6000 FOR N1=1 TO N
6100 FOR N1=1 TO N
6200 FOR N1=1 TO N
6300 FOR N1=1 TO N
6400 FOR N1=1 TO N
6500 FOR N1=1 TO N
6600 FOR N1=1 TO N
6700 FOR N1=1 TO N
6800 FOR N1=1 TO N
6900 FOR N1=1 TO N
7000 FOR N1=1 TO N
7100 FOR N1=1 TO N
7200 FOR N1=1 TO N
7300 FOR N1=1 TO N
7400 FOR N1=1 TO N
7500 FOR N1=1 TO N
7600 FOR N1=1 TO N
7700 FOR N1=1 TO N
7800 FOR N1=1 TO N
7900 FOR N1=1 TO N
8000 FOR N1=1 TO N
8100 FOR N1=1 TO N
8200 FOR N1=1 TO N
8300 FOR N1=1 TO N
8400 FOR N1=1 TO N
8500 FOR N1=1 TO N
8600 FOR N1=1 TO N
8700 FOR N1=1 TO N
8800 FOR N1=1 TO N
8900 FOR N1=1 TO N
9000 FOR N1=1 TO N
9100 FOR N1=1 TO N
9200 FOR N1=1 TO N
9300 FOR N1=1 TO N
9400 FOR N1=1 TO N
9500 FOR N1=1 TO N
9600 FOR N1=1 TO N
9700 FOR N1=1 TO N
9800 FOR N1=1 TO N
9900 FOR N1=1 TO N
10000 FOR N1=1 TO N

```

278 HIT ANY KEY = "PULSE CUAL-
QUER TECLA"
988 HIGH = "MAXIMO"
990 PLAY AGAIN? = "¿VUELVE A
JUGAR?"

BCS

Salta si acarreo activado

Operación: Salta si C = 1

(Ref.: 4.1.3.4)

N Z C I D V

- - - - -

Modo de Direc.	Formato en ensamblador	Código Operan.	Num. Bytes	Num. Ciclos
RELATIVO	BCS Oper.	B0	2	2*

* Suma 1 si se salta a la misma página.

* Suma 2 si se salta a otra página.

BCS

Compara los bits de memoria con acumulador

Operación: AAM, M7 → N, M₆ → V

Los bits 6 y 7 se transfieren al registro de estado. Si el resultado de AAM es cero entonces Z = 1, sino Z = 0.

(Ref.: 4.2.1.1)

N Z C I D V

M7* V - - M₆

Modo de Direc.	Formato en ensamblador	Código Operan.	Num. Bytes	Num. Ciclos
Reg. Caro Absoluto	BIT Oper. BIT Oper.	34 2C	2 3	3 4

BIT**BEQ**

Salta si el resultado es cero

Operación: Salta si Z = 1

(Ref.: 4.1.3.5)

N Z C I D V

- - - - -

Modo de Direc.	Formato en ensamblador	Código Operan.	Num. Bytes	Num. Ciclos
RELATIVO	BEQ Oper.	P0	2	2*

* Suma 1 si se salta a la misma página.

* Suma 2 si se salta a otra página.

BMI

Salta si el resultado es negativo

Operación: Salta si N = 1

(Ref.: 4.1.3.1)

N Z C I D V

- - - - -

Modo de Direc.	Formato en ensamblador	Código Operan.	Num. Bytes	Num. Ciclos
Relativo	BMI Oper.	30	2	2*

* Suma 1 si se salta a la misma página.

* Suma 2 si se salta a otra página.

BMI

VIC-20

Microprocesador: 6502 de MOS TECHNOLOGY de 8 bits.

Memoria: 5 Kbytes de RAM ampliables a 32 K. 20 Kbytes de ROM ampliables a 38 K.

Pantalla: 23 líneas de 22 caracteres. Modulador para conectar a un televisor normal. Señal monóculo video. Colores: 8 para el monitor, 16 para el fondo de la pantalla y 8 para los caracteres individuales. Vídeo inverso. Gráficos: Son gráficos por teclado y alta resolución, por definición del generador de caracteres almacenado en RAM. Definición de 176 por 184 puntos. Teclado: Tipo QWERTY de 62 teclas más control de función definibles por el usuario.

Sonido: Tres voces de tres octavas cada una, decodificadas una octava entre sí, resultando una extensión total de cinco voces. Un generador de onda alfanumérica afinable para efectos especiales, un control general de volumen.

Programas: Lenguaje BASIC, intérprete residente en ROM de 8K. Posibilidad de interpretar los programas en Basic para crear nuevos programas "a medida". El Basic del Vic es uno de los mejores actualmente en el mercado.

Complementos: Port de usuario de 8 bits controlada por software de software de software.

Bus de expansión para ampliaciones de memoria y periféricos.

Port de juegos con conexión para dos procesadores (quadrant), y una pantalla de juegos (quadrant). Almacenamiento de masas Unidad de cassette. CM de disco especial para registrar programas y datos. Ampliación de memoria: En caso de ser necesario ampliar más de un carácter al mismo tiempo, está disponible un módulo VIC 1520 que permite la extensión simultánea de hasta seis caracteres.

VIC 1541 UNIDAD DE DISCO

Capacidad total: 134400 bytes por disco.

Necesidad: 158400 bytes por disco.

Entrada de direcciones: 194 por disco.

Secciones por pista: De 17 a 21.

Bytes por sector: 256.

Pistas: 35.

Bloques: 480 4044 bloques libred.

Soportes de información: Discos standar de 5 1/4 pulgadas, de una sola cara y densidad simple.

Sistema operativo: DOS de COMMODORE integrado (tiene procesador propio y no ocupa memoria del ordenador central).

VIC 1525 IMPRESORA

Método de impresión: Método de 5 x 1 puntos, impreso por un solo martillo.

Modo caracteres: Mayúsculas y minúsculas, símbolos, números y caracteres gráficos del VIC 20.

Modo gráfico: Puntos direccionales del mouse. Seis puntos verticales por columna, 480 columnas máximas. Velocidad: 30 caracteres/segundo, de izquierda a derecha, unidireccional.

Características: Máxima 80. Posibilidad de impresión en doble ancho.

Espaciado entre líneas: 6 líneas/pulgada — modo caracteres, 9 líneas/pulgada — modo gráfico.

Alimentación de papel: Avanzar por tractor.

Ancho de papel: Entre 4,5 y 10 pulgadas.

Copias: Original más dos copias.

CARTUCHOS

Avenda programador: Facilita la edición y depuración de programas en Basic. Instrucciones y comandos: RENAME, MERGE, FIND, CHANGE, DELETE, AUTO, TRACE, STEP, OFF, KEY, EDIT, PROG, DUMP, HELP y KILL.

Super cargador: Introduce el Basic del VIC para reducir considerablemente las instrucciones y comandos en aplicaciones gráficas de sonido y juegos. Instrucciones y comandos: KEY, GRAPHIC, COLOR, POINT, REGION, DRAW, CIRCLE, PAINT, CLEAR, SENSE, SOUND, BOE, COLOR, REDIT, RPOE, ROPEN, RKEY y REND.

Monitor de longitud múltiple: Facilita enormemente la depuración de programas en lenguaje máquina, al cual como complemento del Basic para indicar y poner en marcha rutinas de alta velocidad y manejo de datos en tiempo real. Instrucciones y comandos: ASSEMBLE, BREAKPOINT, DISASSEMBLE, ENABLE, VIRTUAL ZERO PAGE, TELL MEMORY, GO, JUMP, INTERPRET, JUMP TO SUBROUTINE, LOAD MEMORY, NUMBER, QUICK TRACE, REGISTERS, RENAME BREAKPOINTS, SAVE, TRANSFER, WALK y EXIT TO BASIC. Además, incluye control de amplitud de memoria de 5.8 y 95 Kbytes.

CURSO DE INTRODUCCIÓN AL BASIC PARTE I y II.

En forma de libro se ha editado la primera y segunda parte de un curso de Basic que parte "de cero" y está basado en el VIC 20. Van acompañados de dos cintas con programas y ejercicios para autocorrección.

PLOTTER VIC 1520

Método de impresión: Dibuja mediante bolígrafos de colores expresos.

Colores: 4 colores: negro, azul, verde y rojo con cambio desde programa.

Caracteres: Puntos X Y tipo tambor.

Velocidad de impresión: Método de 34 por píxel.

Caracteres por línea: Máxima 80 caracteres, formato de 80, 40, 30 y 10 caracteres.

Juego de caracteres: 96.

Velocidad de dibujo: 764 puntos/seg.

Longitud del paso: 0,2 mm. en dirección X e Y. Velocidad de dibujo de línea: 12,8 mm/seg. en dirección X e Y. 13 mm/seg. en una línea a 45 grados.

Área de dibujo: 480 pines (96 mm) en dirección X. Programado en dirección Y (640 pines). — 999 de una sola vez.

Papel: Rollos de 4,5 pulgadas (114 mm).

MONITOR EN COLOR C-1201

Pantalla: 15 pulgadas (380 mm).

Capacidad de representación: 25 líneas de 48 caracteres.

Resolución: 320 líneas horizontales.

Compatibilidad: VIC 20 y COMMODORE 64.

Conecta: a un regulador de vídeo.

Amplificador y altavoz: Incorporados.

commodore
COMPUTER



microelectrónica
y control, s.a.

Tecnoigra Serra, 7-5. Tel. 250 51 03. BARCELONA-29
Ponessa, 47 3° G. Tel. 248 95 70. MADRID-8

